Spectral Element Method for Flow Simulation

Paul Fischer

University of Illinois
Departments of Computer Science and
Mechanical Science and Engineering

Mathematics and Computer Science Division
Argonne National Laboratory, U.S.A.

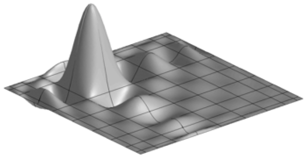
fischerp@illinois.edu

www.mcs.anl.gov/~fischer/sem

Introduction

- The spectral element method (SEM) is a *high-order* weighted residual technique in which the computational domain is tessellated into
  - *curvilinear squares or triangles* in 2D, or
  - *curvilinear bricks or tetrahedra* in 3D.

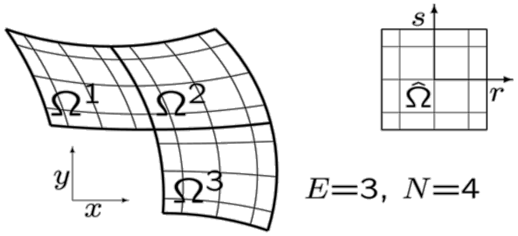
- Within each of these elements (squares, bricks, etc.) the solution is represented by *N*th-order polynomials, where *N*=5-15 is most common but N=1 to 100 or beyond is feasible.



2D basis function, N=10

$$u(x,y)|_{\Omega^e} = \sum_{i=0}^{N}\sum_{j=0}^{N} u^e_{ij}\, h_i(r)\, h_j(s)$$

$$h_i(r) \in \mathcal{P}_N(r), \qquad h_i(\xi_j) = \delta_{ij}$$



SEM & Transport Phenomena

- The main advantage of the SEM is manifest in transport problems that are characterized by *first-order* differential operators in space, e.g.

$$\text{Advection:} \qquad \frac{\partial u}{\partial t} + \mathbf{c}\cdot\nabla u = 0 \qquad (1)$$

$$\text{Advection-Diffusion:} \qquad \frac{\partial u}{\partial t} + \mathbf{c}\cdot\nabla u = \nu\nabla^2 u \qquad (2)$$

$$\text{Navier-Stokes:} \qquad \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla \mathbf{u} = -\nabla p + \nu\nabla^2 u \qquad (3)$$

$$\nabla\cdot\mathbf{u} = 0$$

- In nondimensional form, we have $|u|$ ~1, ν = 1/Pe for (2) and ν =1/Re for (3), respective inverse Peclet and Reynolds numbers, which are small (e.g., 10^{-4}-10^{-6}) for most engineering problems.

- Such problems are characterized by *minimal dissipation* →
The solution propagates for long times with minimal decay or energy loss.

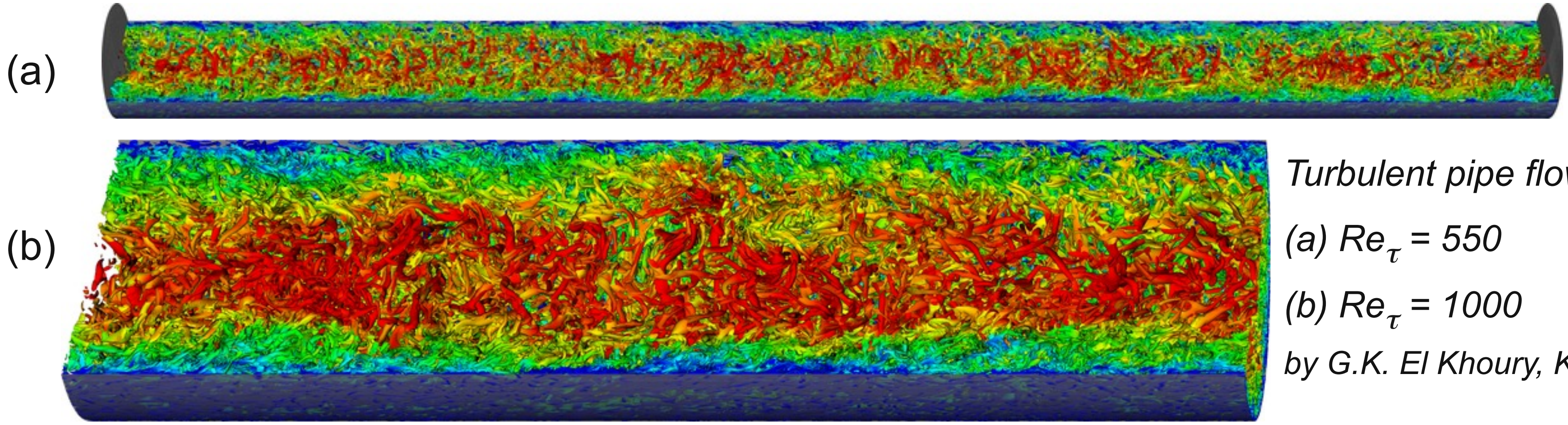
SEM & Transport Phenomena

- These problems are particularly challenging because, unlike diffusion, where

$$\frac{\partial u}{\partial t} = \nu \nabla^2 u \quad \longrightarrow \quad \hat{u}_k(t) \sim e^{-\nu k^2 t}$$

implies rapid decay of high wavenumber (k) components (and errors), the high-k components and errors in advection-dominated problems *persist.*

- Turbulence provides a classic example of this phenomena:



Turbulent pipe flow:

(a) Re_τ = 550

(b) Re_τ = 1000

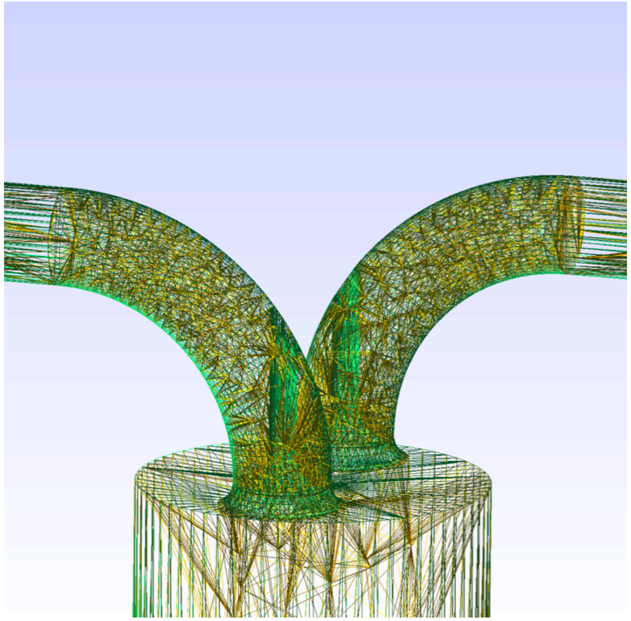
by G.K. El Khoury, KTH

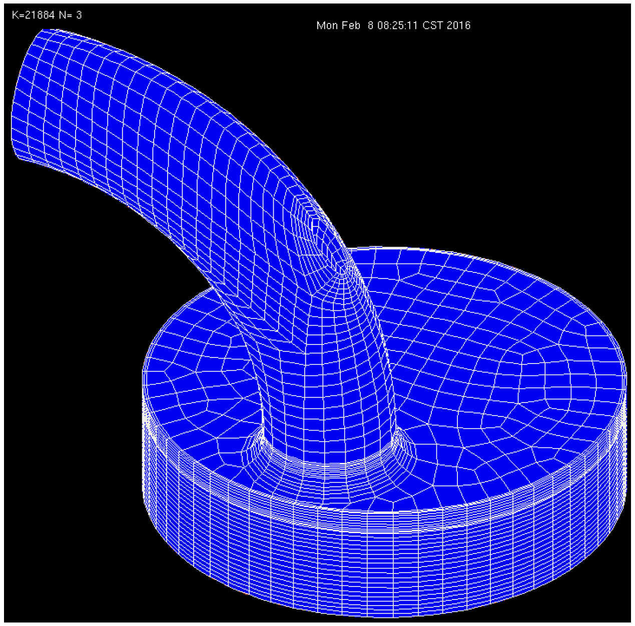
Turbulence in an IC Engine

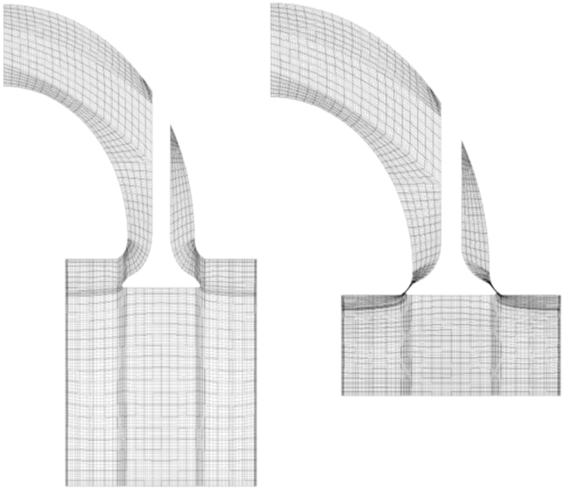
Starting with an .stl file, mesh is made with CUBIT.

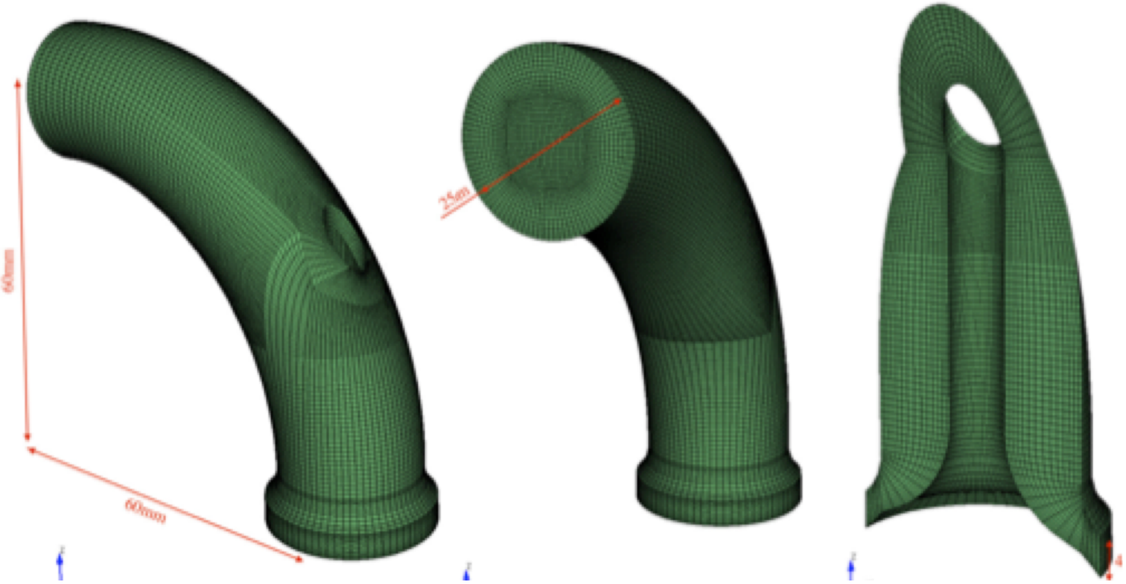
The lower panel shows the mesh motion.

Lower-right shows a very fine mesh used for the intake port.



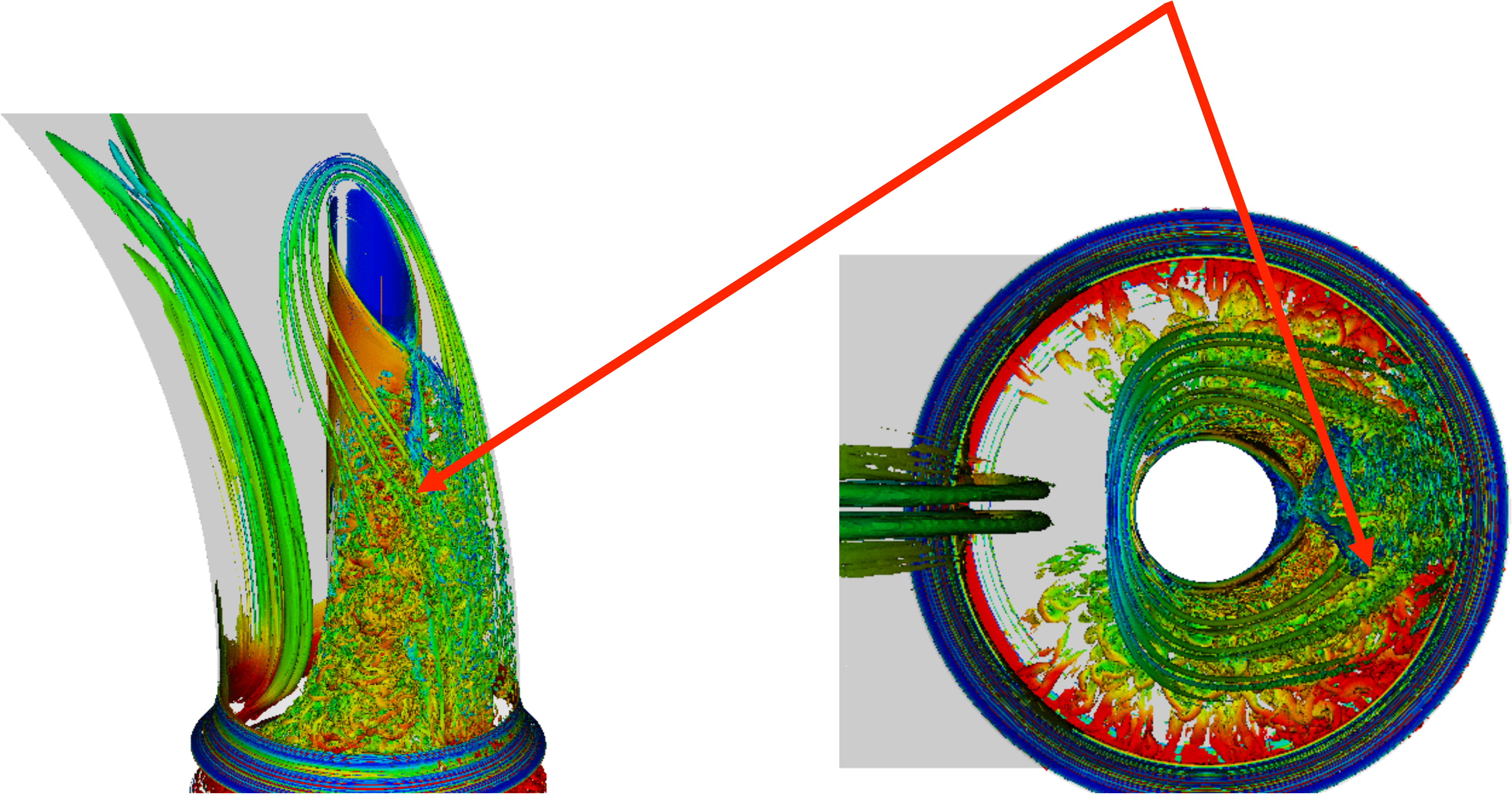






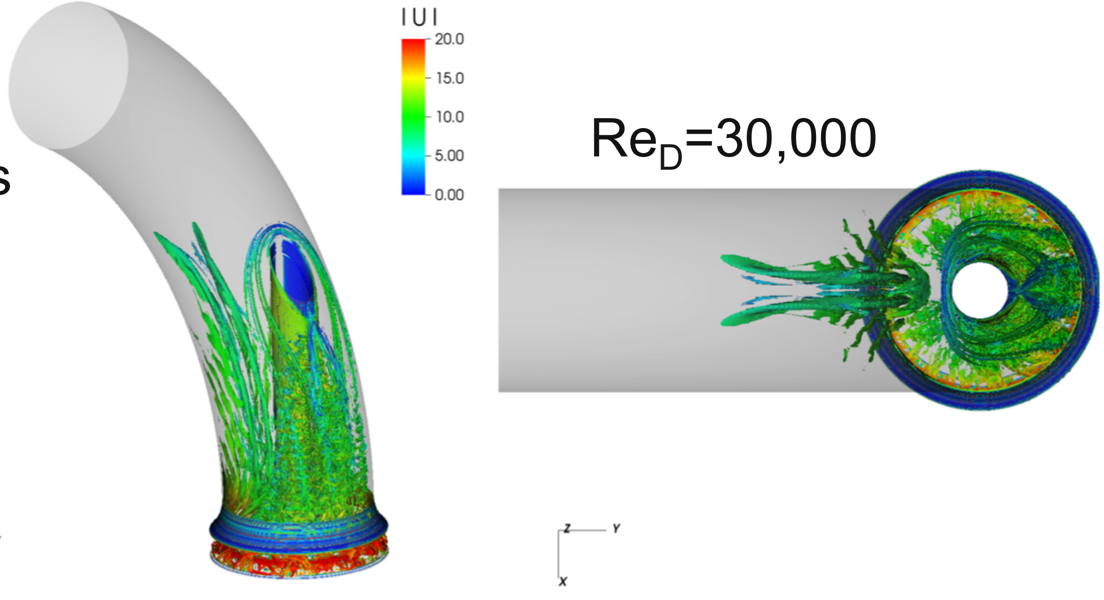
Vortex Breakdown at Re_D = 15,000

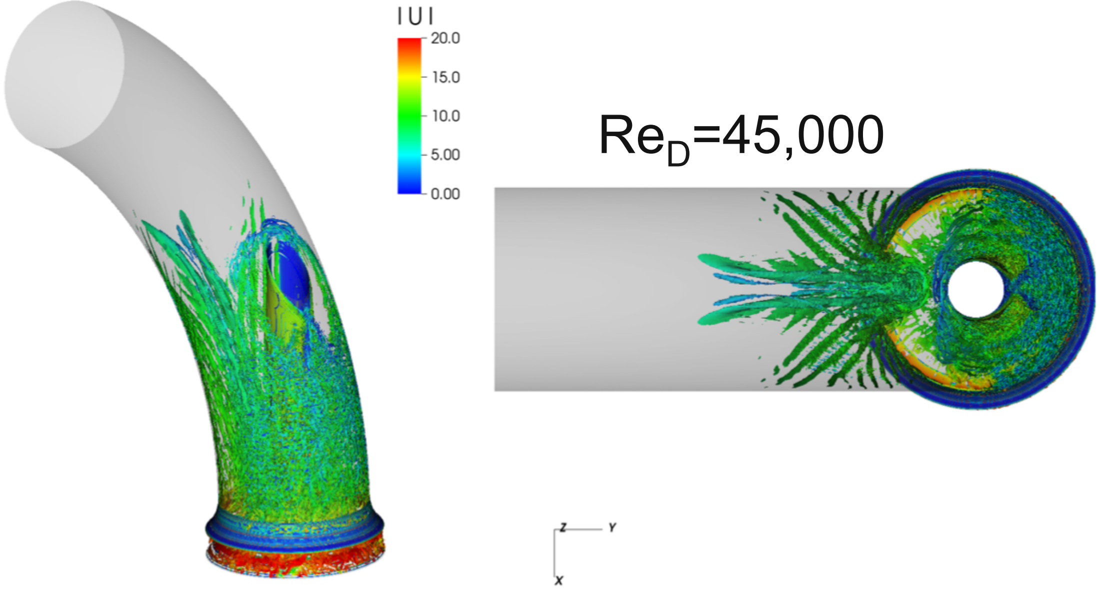
- **These are extremely well resolved calculations performed on Mira.**
- Note the highly-resolved filamental horseshoe vortices around the base of the valve stem that ultimately break down into a hairpin vortex chain.



Influence of Reynolds Number

- The Reynolds number has a significant impact on the scales of motion.
- The Reynolds number in the intake port of the TCC engine peaks at around Re=45000 at 670 RPM.
- The Reynolds number in the combustion chamber is about Re=15000.

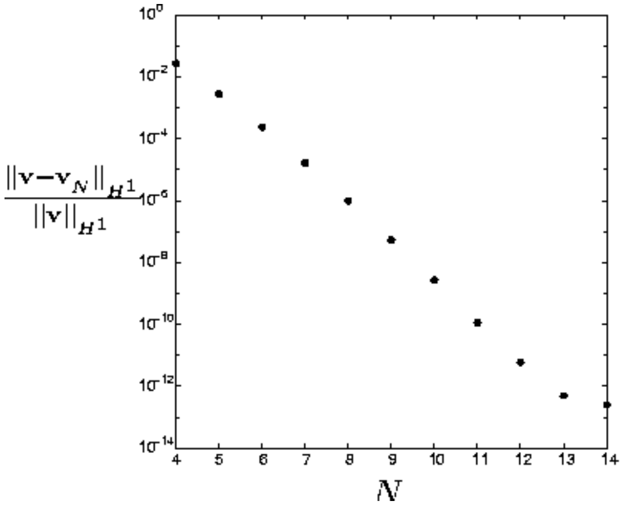

Re_D=30,000

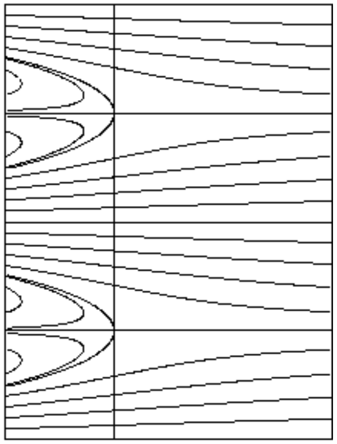

Re_D=45,000

Spectral Element Method: Exponential Convergence

Exact Navier-Stokes Solution (Kovazsnay ‘48)

- *4 orders-of-magnitude error reduction when doubling the resolution in each direction*

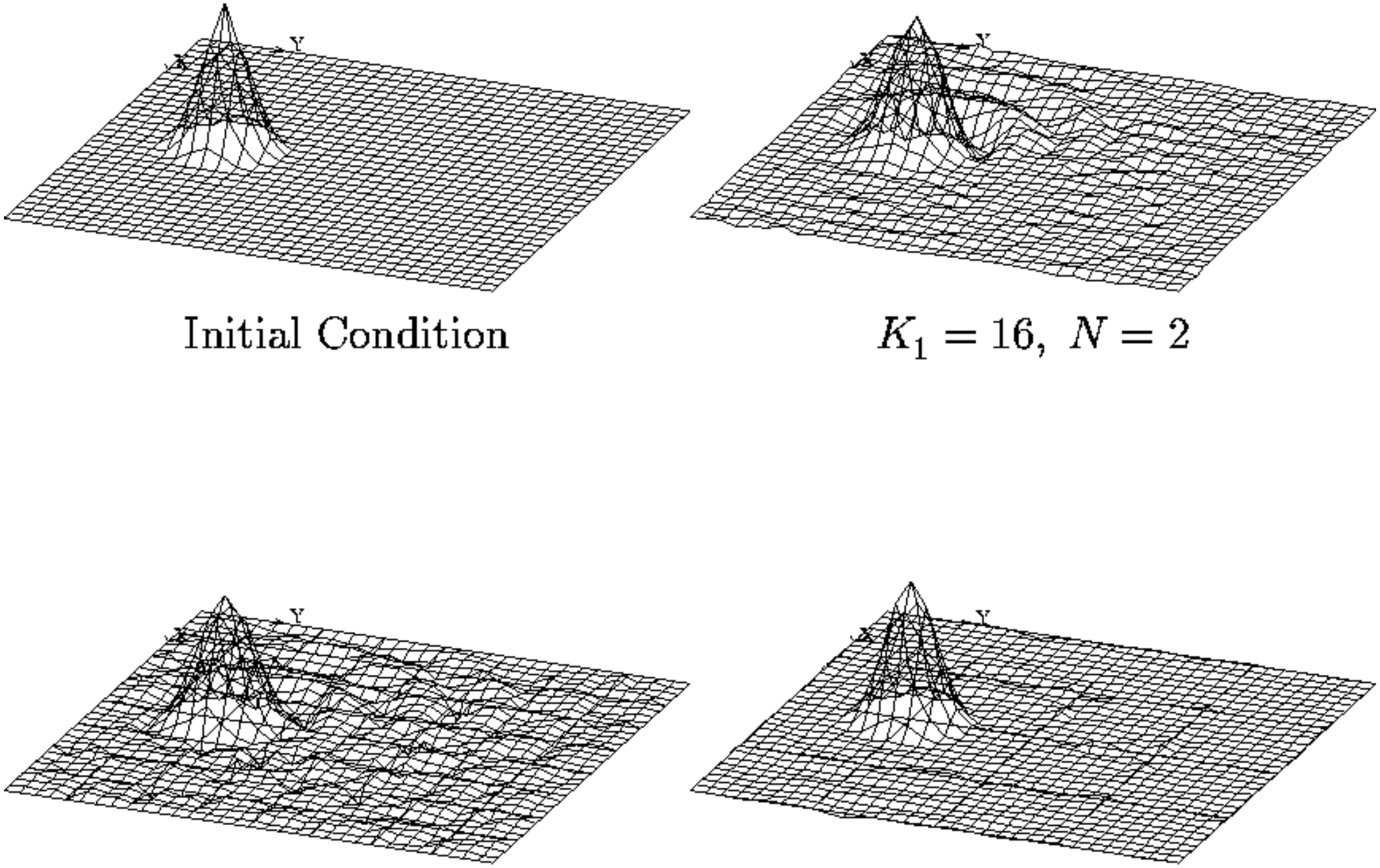




- *For a given error,*
  - *Reduced number of gridpoints*
  - *Reduced memory footprint.*
  - *Reduced data movement.*

$$
\begin{aligned}
v_x &= 1 - e^{\lambda x} \cos 2\pi y \\
v_y &= \frac{\lambda}{2\pi} e^{\lambda x} \sin 2\pi y \\
\lambda &:= \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}
\end{aligned}
$$

The SEM provides excellent transport properties, even for non-smooth *solutions*



Initial Condition

$K_1 = 16,\ N = 2$

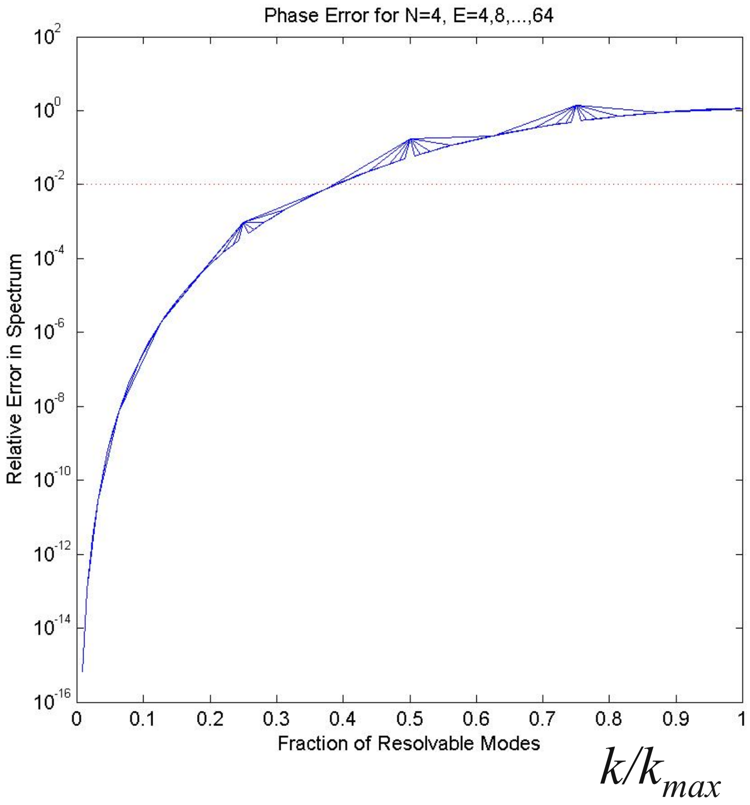
$K_1 = 8,\ N = 4$

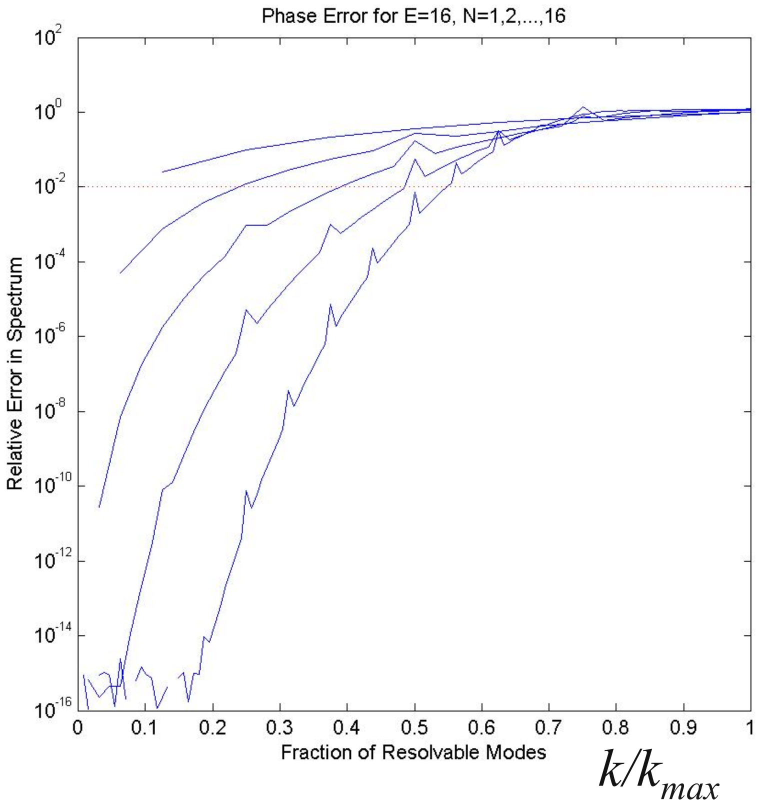
$K_1 = 4,\ N = 8$

Convection of non-smooth data on a 32x32 grid (K$_1$ x K$_1$ spectral elements of order N).

(cf. Gottlieb & Orszag 77)

Relative Phase Error for h *vs.* p *Refinement:* $u_t + u_x = 0$





- $k_{max} := n/2$

- Fraction of *accurately resolved* modes is increased *only* through increasing order (N or p)

Influence of Scaling on Discretization

Large problem sizes enabled by peta- and exascale computers allow propagation of small features (size λ) over distances L >> λ. If speed ~ 1, then $t_{final} \sim L/\lambda$.

- Dispersion errors accumulate linearly with time:

  ~|correct speed – numerical speed| * t *(for each wavenumber)*

  → error_{t_final} ~ (L / λ) * | numerical dispersion error |

- For fixed final error ε_f, require: numerical dispersion error ~ $(\lambda / L)\varepsilon_f$, << 1.

- *We want methods with low dispersion error!*

High-order methods can efficiently deliver small dispersion errors.

(Kreiss & Oliger 72, Gottlieb et al. 2007)

Linear Advection Example

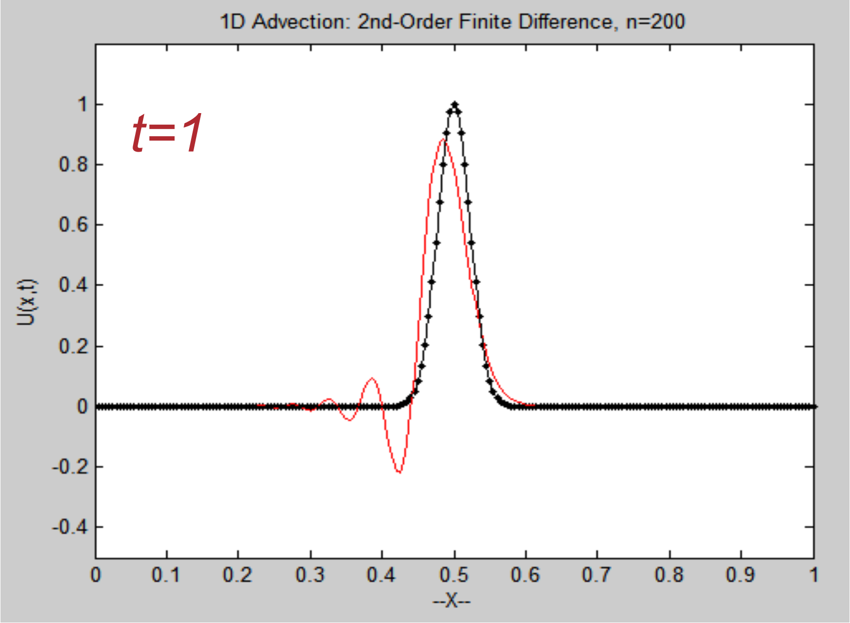
- Here, we consider linear advection with periodic BCs on [0,1]:

$$\frac{\partial u}{\partial t} + c\,\frac{\partial u}{\partial x} = 0, \qquad u(0,t) = u(1,t) \qquad u(x,0) = u_0.$$

- With speed c = 1, the travelling wave solution should return to the initial condition after each unit time.

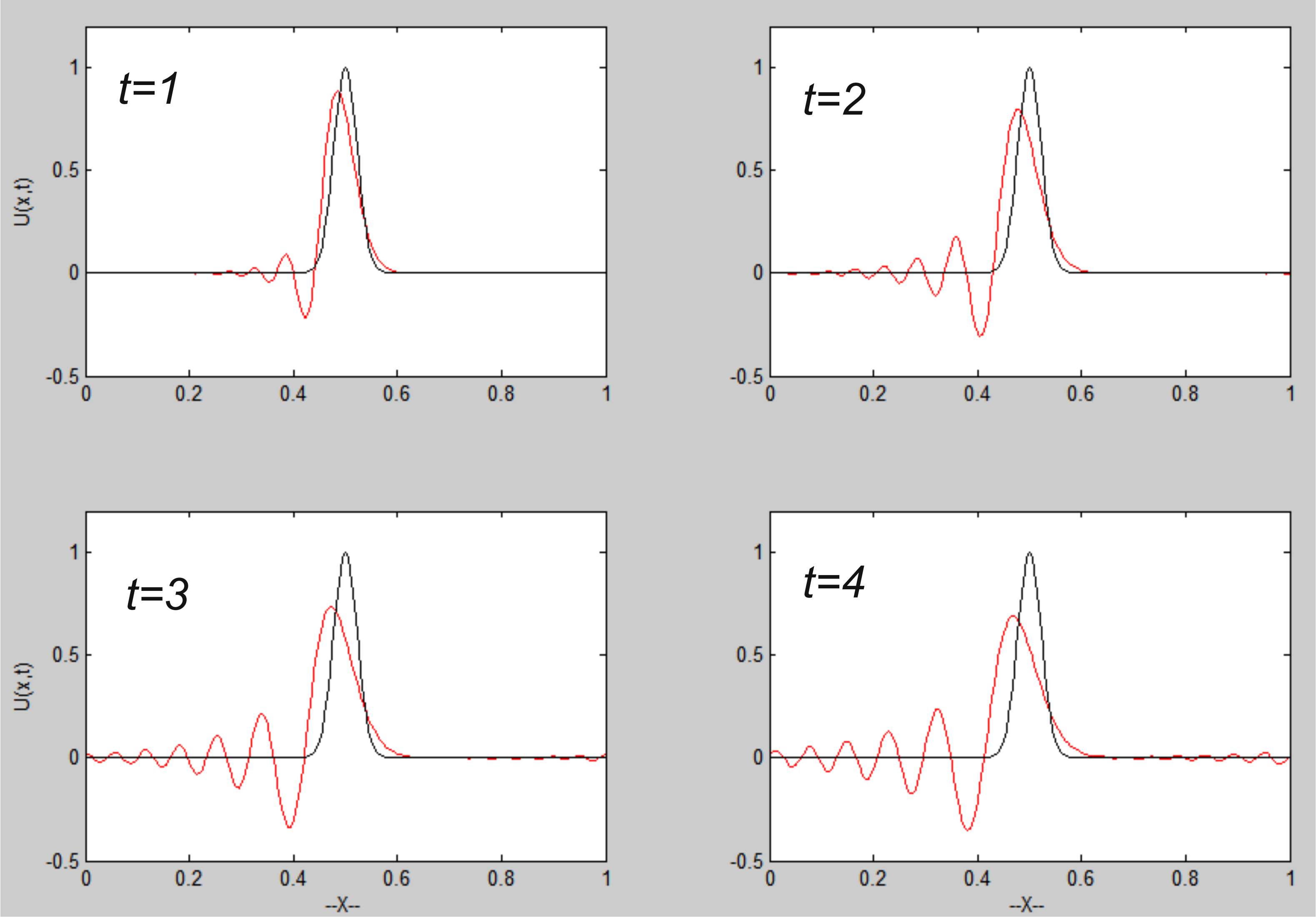
- This result is not realized numerically, especially for low-order discretizations.

- Although the initial condition (black) is well-resolved with n=200 points, the 2nd-order solution exhibits trailing waves (red) even after one revolution.



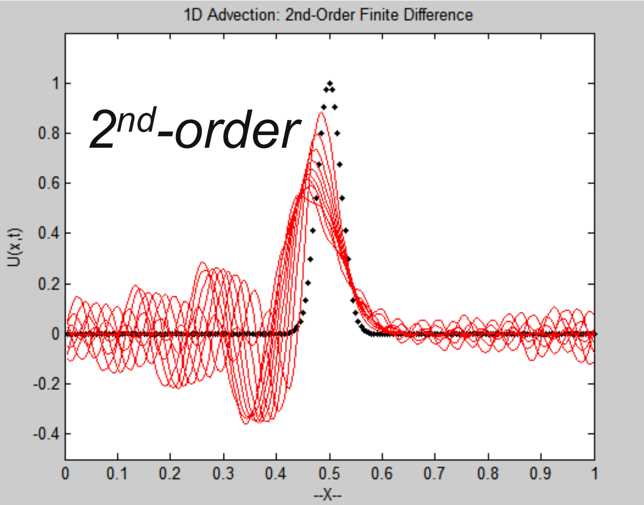
Numerical Dispersion, 2nd-order Spatial Discretization

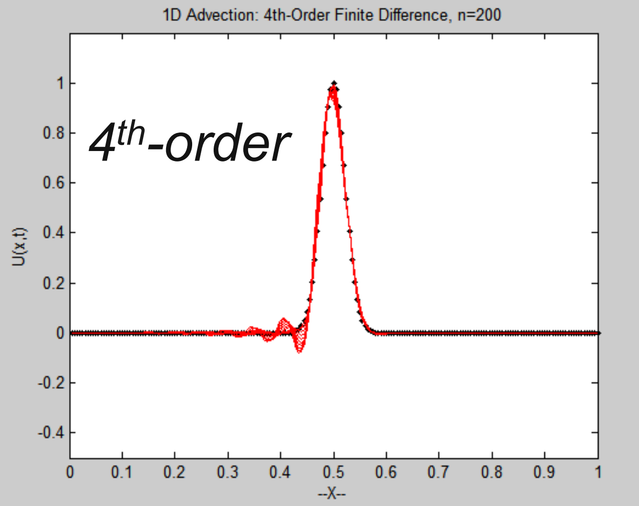
- At later times, the dispersion just becomes worse…



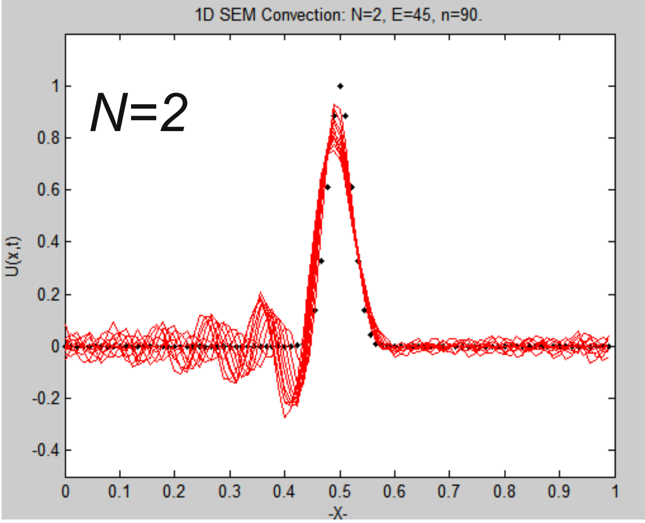
Cumulative Dispersion at t=10 for Varying Order & Resolution

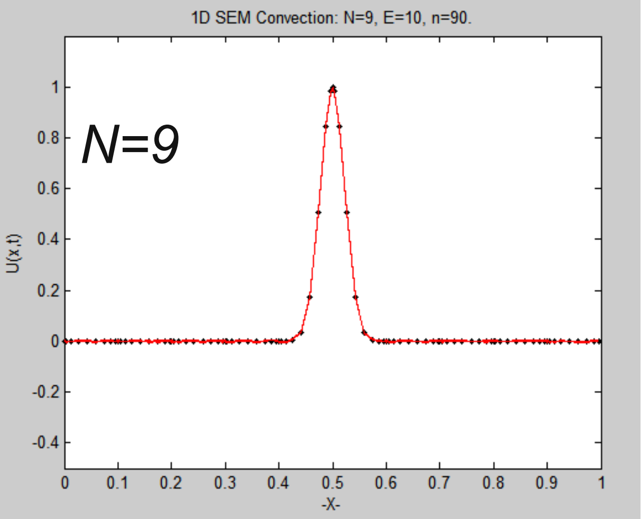
Finite Difference, n=200





SEM, n=90





Computational Savings

- We observe that with only 90 points the 9th-order SEM is able to outperform 4th-order finite differences with 200 points.

- This translates into > 8x reduction in the number of points for problems in 3D.

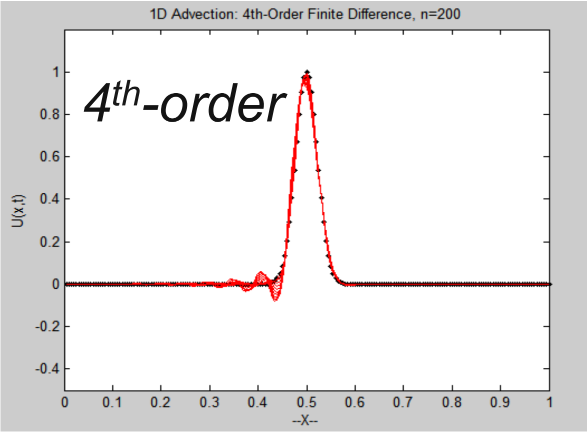
- We will see that the *cost-per-gridpoint* for the two methods is essentially the same, meaning that the SEM offers an order-of-magnitude reduction in computational costs for this class of problems.

Matlab Demos

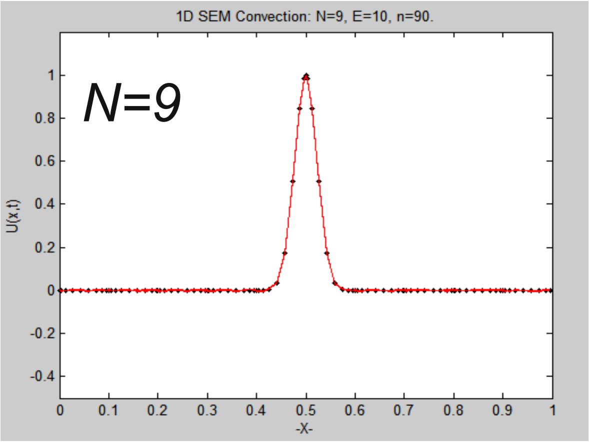
- demo_fd2.m
- demo_fd4.m
- demo_sem90.m

Cumulative Dispersion at t=10 for Varying Order: FD & SEM

Finite Difference, n=200



SEM, n=90



- *The 90 point SEM with N=9 has much less dispersion than 4th-order FD with n=200 points.*

- *This +2X savings in 1D translates into > 8X savings in 3D. (To leading order, cost ~ n.)*

- *Note that one can also go to higher order FD (and there are some advantages over SEM).*

- *However, there are also many advantages (BCs, geometric flexibility) to the SEM.*

SEM Derivation

- We turn now to the heart of the course.

- We will begin with development of the SEM in 1D for the
  - Poisson equation
  - steady convection-diffusion
  - unsteady advection

  These are constituent subproblems in the simulation of incompressible flows.

- We then turn to higher space dimensions, with a primary focus on 2D, for conciseness.

Spectral Element Method: 1D

The SEM is based on the *weighted residual technique*, which is essentially a method of undetermined coefficients.

Let's consider the 1D Poisson equation

$$-\frac{d^2\tilde{u}}{dx^2} = f(x), \qquad \tilde{u}(0) = \tilde{u}(1) = 0.$$

We seek an approximate solution u from a finite-dimensional *trial space* X_0^N,

$$u \in X_0^N := \text{span}\{\phi_1(x), \phi_1(x), \dots, \phi_n(x)\}, \qquad \phi_j(0) = \phi_j(1) = 0.$$

(We use the subscript on X_0^N to indicate that functions in this space satisfy the homogeneous Dirichlet boundary conditions.)

Trial Solution and Residual

The *trial solution* has the form

$$\boxed{u(x) = \sum_{j=1}^{n} \phi_j(x)\hat{u}_j.}$$

The ϕ_j's are the *basis functions.*

The $\hat{u}_j$'s are the *basis coefficients.*

We define the *residual*, $r(x;u) = r(x)$, as

$$r(x) := f(x) + \frac{d^2u}{dx^2}.$$

It is clear that r is some measure of the error given that

$$r \equiv 0 \quad \textit{iff} \quad u = \tilde{u}.$$

(In fact, it is the *only* measure of error available to us.)

Trial Solution and Residual

Another equivalent definition of the residual derives from the fact that

$$f(x) \;\equiv\; -\frac{d^2\tilde{u}}{dx^2}$$

for the exact solution $\tilde{u}(x)$. Substituting, we have,

$$r(x) \;:=\; f(x) + \frac{d^2u}{dx^2} \;=\; -\frac{d^2}{dx^2}\left(\tilde{u} - u\right) \;=\; -\frac{d^2e}{dx^2},$$

where $e(x) := \tilde{u}(x) - u(x)$ is the *error.*

The residual associated with $u(x)$ is thus the differential operator applied to the error function (with homogeneous boundary conditions).

This form will be of value later on.

WRT and Test Functions

In the WRT, we don't require $r \equiv 0$.

Rather, we insist that r be ($\mathcal{L}^2$-) orthogonal to a set of functions v belonging to the the *test space*, Y_0^N,

$$\int_0^1 v\, r\, dx \;=\; 0, \qquad \forall\, v \in Y_0^N .$$

Convergence is attained as we complete the approximation space, that is, as we let $n \longrightarrow \infty$ for a reasonable set of ϕ_js.

It is most common to take the trial and test spaces to be the same, $Y_0^N = X_0^N$, which leads to the *Galerkin* formulation,

Find $u \in X_0^N$ *such that*

$$-\int_0^1 v\, \frac{d^2u}{dx^2}\, dx \;=\; \int_0^1 v\, f\, dx \qquad \forall\, v \in X_0^N .$$

Reducing Continuity to C^0

It appears that u must be twice differentiable.

However, if we integrate by parts, we can reduce the continuity requirements on u.

Let $\mathcal{I}$ denote the l.h.s. of the preceding equation:

$$\begin{aligned}
\mathcal{I} &= -\int_0^1 v \, \frac{d^2u}{dx^2} \, dx \\
&= \int_0^1 \frac{dv}{dx} \, \frac{du}{dx} \, dx \; - \; \cancel{v \frac{du}{dx}\bigg|_0^1} \\
&= \int_0^1 \frac{dv}{dx} \, \frac{du}{dx} \, dx
\end{aligned}$$

For a variety of technical reasons, it's generally a good idea to balance the continuity requirements of v and u, to the extent possible.

Weighted Residual / Variational Formulation

Using the integration-by-parts trick of the preceding slide (the only bit of calculus we'll require), we arrive at the weighted residual statement for u.

$$\textit{Find } u \in X_0^N \textit{ such that}$$

$$\int_0^1 \frac{dv}{dx}\frac{du}{dx}\,dx \;=\; \int_0^1 v\,f\,dx \qquad \forall\, v \in X_0^N.$$

Convergence is attained by taking the limit $n \longrightarrow \infty$ for an appropriate set of basis functions in X_0^N.

Important Properties of the Galerkin Formulation

- An essential property of the Galerkin formulation for the Poisson equation is that the solution is the ***best fit*** in the approximation space, with respect to the energy norm.

  Specifically, we consider the bilinear form,

  $$a(v,u) \;:=\; \int_0^1 \frac{dv}{dx}\frac{du}{dx}\,dx,$$

  and associated semi-norm,

  $$||u||_a^2 \;:=\; a(u,u),$$

  which is in fact a norm for all u satisfying the boundary conditions.

- It is straightforward to show that our Galerkin solution, u, is the closest solution to the exact $\tilde{u}$ in the a-norm. That is,

  $$||\, u - \tilde{u} \,||_a \;\; \leq \;\; ||\, w - \tilde{u} \,||_a \quad \textit{for all } w \in X_0^N$$

- In fact, u is closer to $\tilde{u}$ than the interpolant of $\tilde{u}$.

Best Fit Property, 1/4

Define:

$$\begin{aligned}
\mathcal{L}^2_\Omega &= \left\{ v : \int_\Omega v^2 \, dx < \infty \right\} \\
\mathcal{H}^1 &= \left\{ v : \; v \in \mathcal{L}^2_\Omega, \; \int_\Omega (v')^2 \, dx < \infty \right\} \\
\mathcal{H}^1_0 &= \left\{ v : \; v \in \mathcal{H}^1, \; v|_{\partial\Omega} = 0 \right\}
\end{aligned}$$

Then, $\forall u, v \in \mathcal{H}^1_0$,

$$\begin{aligned}
a(u,v) &:= \int_\Omega u'v' \, dx && (a \text{ inner-product}) \\
||v||_a &:= \sqrt{a(v,v)} && (a\text{-norm}) \\
||\alpha v||_a &= |\alpha| \sqrt{a(v,v)} && \alpha \in \mathbb{R} \\
||v||_a &= 0 \text{ iff } v \equiv 0.
\end{aligned}$$

Best Fit Property, 2/4

We now demonstrate that $||u - \tilde{u}||_a \leq ||w - \tilde{u}||_a \; \forall w \in X_0^N$.

Let $e := u - \tilde{u}$ and $v := w - u \in X_0^N$.

For any $w \in X_0^N$ we have

$$\begin{aligned}
||w - \tilde{u}||_a^2 &= ||v + u - \tilde{u}||_a^2 \\
&= ||v + e||_a^2 \\
&= \int_0^1 (v + e)' \, (v + e)' \, dx \\
&= \int_0^1 (v')^2 dx \; + \; 2 \int_0^1 v' \, e' dx \; + \; \int_0^1 (e')^2 dx
\end{aligned}$$

Best Fit Property, 3/4

The second term vanishes:

$$
\begin{aligned}
\int_0^1 v'\, e' dx + \quad &= \int_0^1 v'\, (u - \tilde{u})' dx \\
&= \int_0^1 v'\, u'\, dx \; - \; \int_0^1 v'\, \tilde{u}'\, dx \\
&= \int_0^1 v'\, u'\, dx \; + \; \int_0^1 v\, \tilde{u}''\, dx \; - \; v\, \tilde{u}' \Big|_0^1 \\
&= \int_0^1 v'\, u'\, dx \; - \; \int_0^1 v\, f\, dx \\
&= 0 \qquad \forall\, v \in X_0^N . \qquad \text{(by def'n of } u\text{)}
\end{aligned}
$$

0 because of BCs

Best Fit Property 4/4

In summary, for any $w \in X_0^N$ we have

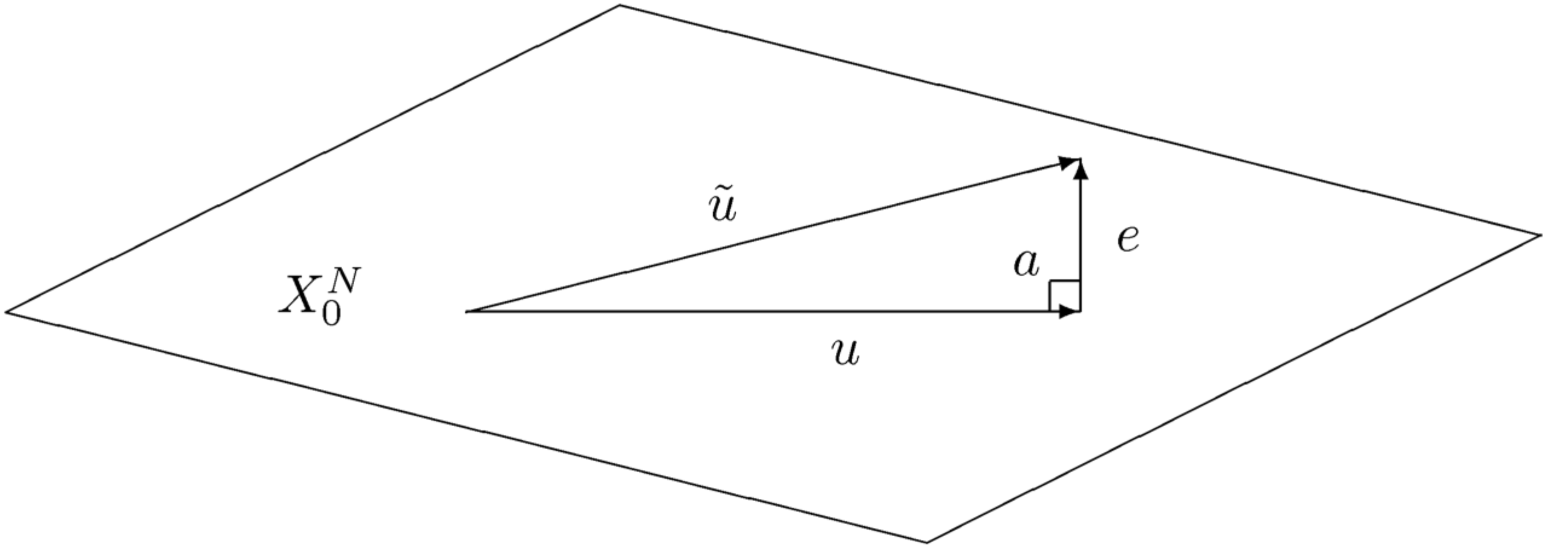
$$\begin{aligned} ||w - \tilde{u}||_a^2 &= ||v + u - \tilde{u}||_a^2 \\ &= ||v + e||_a^2 \\ &= \int_0^1 (v')^2 dx \; + \; 2 \int_0^1 \cancel{v' \, e' dx} \; + \int_0^1 (e')^2 dx \\ &= \int_0^1 (v')^2 dx \; + \int_0^1 (e')^2 dx \\ &\geq \int_0^1 (e')^2 dx \; = \; ||u - \tilde{u}||_a^2 \end{aligned}$$

Thus, of *all* functions in X_0^N, u is the *closest* to $\tilde{u}$ in the a-norm.

A deeper analysis establishes that, for $\tilde{u}$ analytic, one has for the spectral element method

$$||\tilde{u} - u||_{\mathcal{H}_0^1} \;\leq\; C\, e^{-\gamma N}$$

Best Fit Viewed as a Projection



- Note that this result also demonstrates that $a(v, e) = 0$ for all $v \in X_0^N$.
- That is, the Galerkin statement is equivalent to having the *error*,

$$e := u - \tilde{u} \perp_a X_0^N.$$

- Thus, u is the *projection* of $\tilde{u}$ onto X_0^N in the a inner-product.
- The procedure is often referred to as a Galerkin projection.

Formulation of the Discrete Problem

- Up to now, we have dealt with abstract issues and have established the important best-fit property.

- From here on, we move to more practical issues.

Formulation of the Discrete Problem

We can now easily generate our discrete system that allows us to compute the set of basis coefficients. Let

$$\begin{aligned} \underline{u} &:= (u_1\, u_2\, \ldots\, u_n)^T, \\ \underline{v} &:= (v_1\, v_2\, \ldots\, v_n)^T. \end{aligned}$$

Then

$$\begin{aligned} \mathcal{I} := \int_\Omega v'u'\,dx &= \int_\Omega \left(\sum_{i=1}^n \phi_i'(x) v_i \right) \left(\sum_{j=1}^n \phi_j'(x) u_j \right) dx \\ &= \sum_{i=1}^n \sum_{j=1}^n v_i \left(\int_\Omega \phi_i'(x)\phi_j'(x)\,dx \right) u_j \\ &= \sum_{i=1}^n \sum_{j=1}^n v_i\, A_{ij}\, u_j, \;=\; \underline{v}^T A \underline{u}, \end{aligned}$$

with the (global) *stiffness matrix*, A, given by

$$A_{ij} \;:=\; \int_\Omega \phi_i'(x)\phi_j'(x)\,dx.$$

Formulation of the Discrete Problem

We proceed in a similar way with the right-hand side. Assuming

$$f(x) \quad = \quad \sum_{j=0}^{n} \phi_j(x) f_j$$

(which is *way* overly restrictive, since $f \in \mathcal{L}^2_\Omega$ suffices), then

$$\begin{aligned}
\mathcal{I} \;=\; \int_\Omega v f \, dx \quad &= \quad \left(\sum_{i=1}^{n} \phi_i(x) v_i \right) \left(\sum_{j=1}^{n} \phi_j'(x) f_j \right) \, dx \\
&= \quad \sum_{i=1}^{n} \sum_{j=1}^{n} v_i \, \left(\int_\Omega \phi_i(x) \phi_j(x) \, dx \right) \, f_j \\
&= \quad \sum_{i=1}^{n} \sum_{j=1}^{n} v_i \, B_{ij} \, f_j, \;=\; \underline{v}^T B \underline{f},
\end{aligned}$$

with the (global) *mass matrix*, B, given by

$$B_{ij} \quad := \quad \int_\Omega \phi_i(x) \phi_j(x) \, dx.$$

Formulation of the Discrete Problem

Combining the results of the two previous slides, we have:

$$\mathcal{I} \;=\; \underline{v}^T A \underline{u} = \underline{v}^T B \underline{f} \qquad \forall \underline{v} \in \mathbb{R}^n,$$

which implies

$$A\underline{u} = B\underline{f}.$$

Since A is symmetric positive definite, this system is *solvable*.

Choice of Spaces & Bases

- *At this point, it's time to get specific and choose the **space**, X_0^N, and associated **basis**, $\{ \phi_i \}$.*

- *The former influences convergence, i.e.,*
  - *How large or small n must be for a given error.*

- *The latter influences implementation, i.e.,*
  - *details and level of complexity, and*
  - *performance (time to solution, for a given error).*

- *Keep in mind that our goal is to solve high Re / Pe flow problems, so the convergence question is driven by considerations in the convection-dominated limit.*

- *Interestingly, for incompressible or low Mach-number flows, the performance question is largely driven by the pressure-Poisson equation, which governs the fastest time-scale in the problem.*

Incompressible Navier-Stokes Equations

$$
\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} \\
\nabla \cdot \mathbf{u} &= 0
\end{aligned}
$$

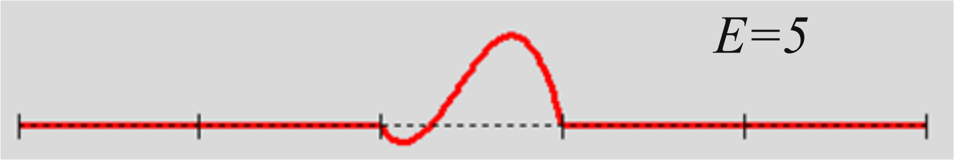
Reynolds number Re > ~1000

- *small amount of diffusion*
- *highly nonlinear* (small scale structures result)

Must discretize in space and time…

Spaces and Bases for the SEM

- *For the spectral element method in R^1, we choose X^N to be the space of piecewise polynomials of degree N on each element, Ω^e, e=1,…,E. For example:*



- *Within each element, one has a choice between modal or nodal bases.*
- *The choice is largely immaterial because of the best-fit property.*
- *It is easy to convert from modal to nodal and back, provided that both representations are **stable.***
- *So, within a given code, we might alternate between representations, depending on the operation at hand.*

Unstable and Stable Bases within the Elements

- *Examples of* ***unstable*** *bases are:*
  - Monomials (modal): $\phi_i = x^i$
  - High-order Lagrange interpolants (nodal) on *uniformly-spaced* points.

- Examples of *stable* bases are:
  - Orthogonal polynomials (modal), e.g.,
    - *Legendre polynomials:* $L_k(x)$, *or*
    - *bubble functions:* $\phi_k(x) := L_{k+1}(x) - L_{k-1}(x)$.
  - Lagrange (nodal) polynomials based on Gauss quadrature points (e.g., Gauss-Legendre, Gauss-Chebyshev, Gauss-Lobatto-Legendre, etc.)

- For the SEM, we typically use nodal bases on the Gauss-Lobatto-Legendre (GLL) quadrature points. However, we often map back and forth between GLL-based nodal values and Legendre or bubble function modal bases, *with minimal information loss*.

Aside: GLL Points and Legendre Polynomials

The GLL points are the zeros of $(1-x^2)\,L'_N(x)$.

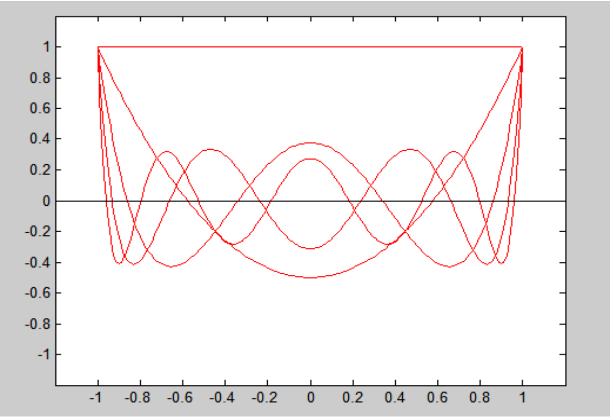
The Legendre polynomials are orthogonal with respect to the L^2 inner product,

$$\int_{-1}^{1} L_i(x)\,L_j(x)\,dx \;=\; \delta_{ij}, \qquad L_i(x) \in \mathbb{P}_i.$$

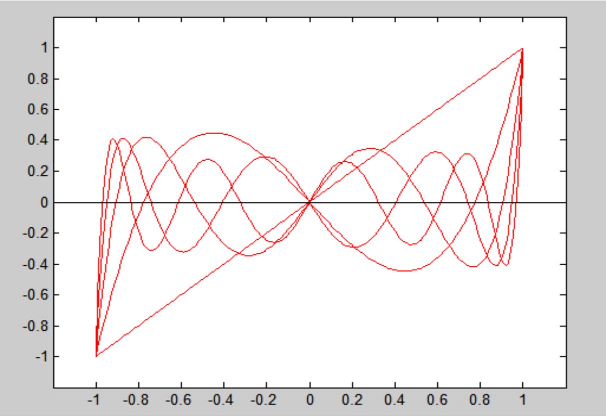
They can be efficiently and stably computed using the 3-term recurrence,

$$\begin{aligned} &L_0(x) := 1, \quad L_1(x) = x, \\ &L_k(x) = \frac{1}{k}\left[(2k-1)\,x\,L_{k-1}(x) \;-\; (k-1)\,L_{k-2}(x)\right]. \end{aligned}$$

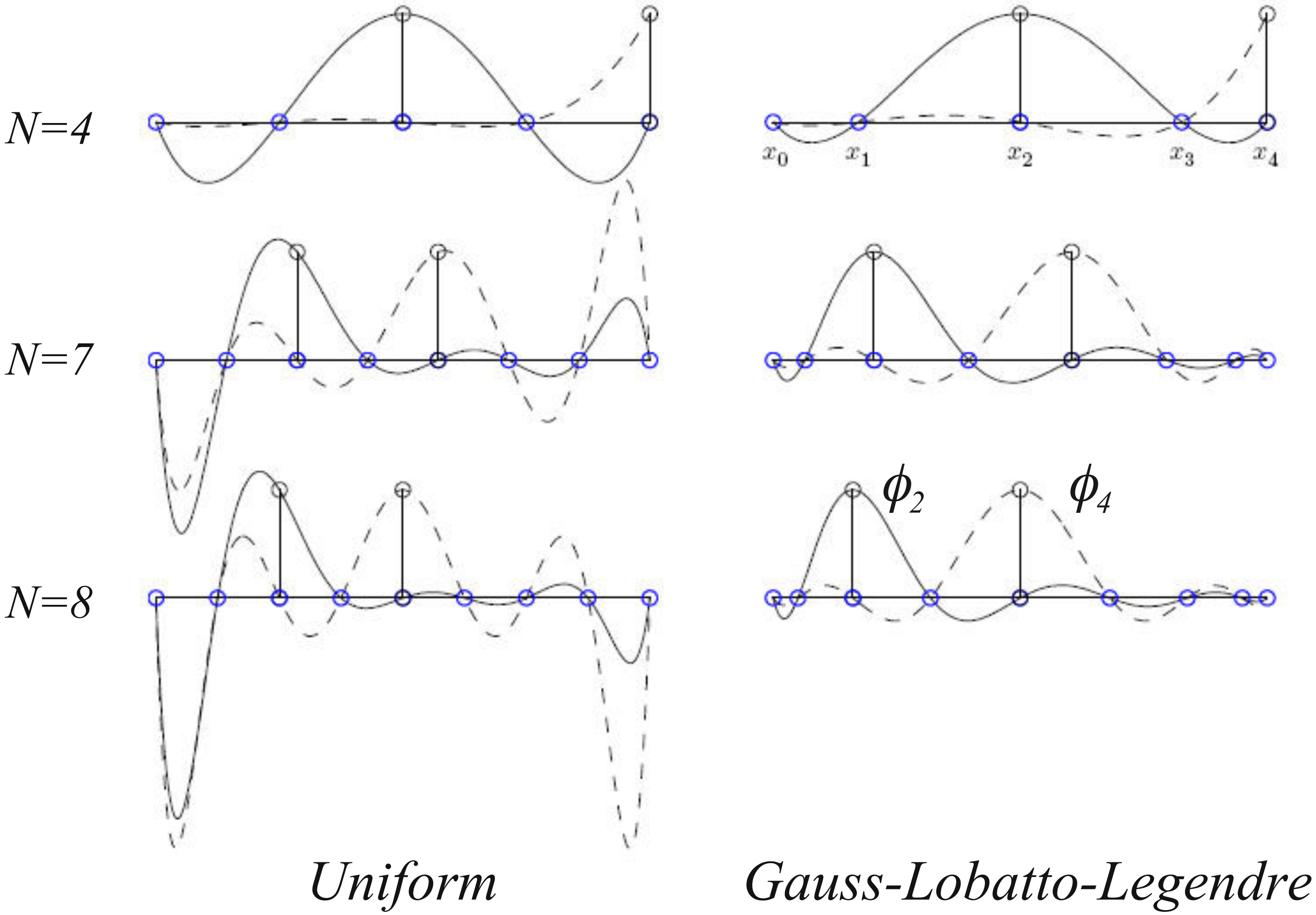
Even Legendre Polynomials, L_0–L_8



Odd Legendre Polynomials, L_1–L_9



Lagrange Polynomials: Good and Bad Point Distributions



Piecewise Polynomial Bases: Linear and Quadratic

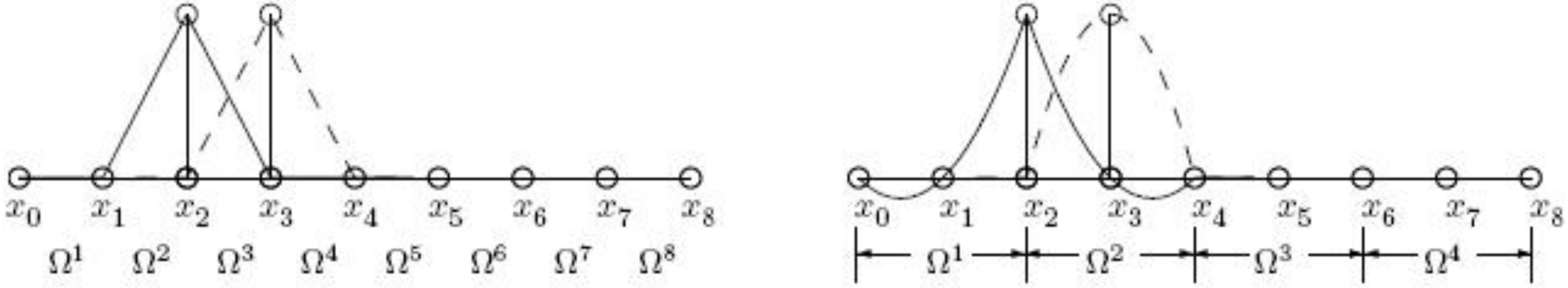
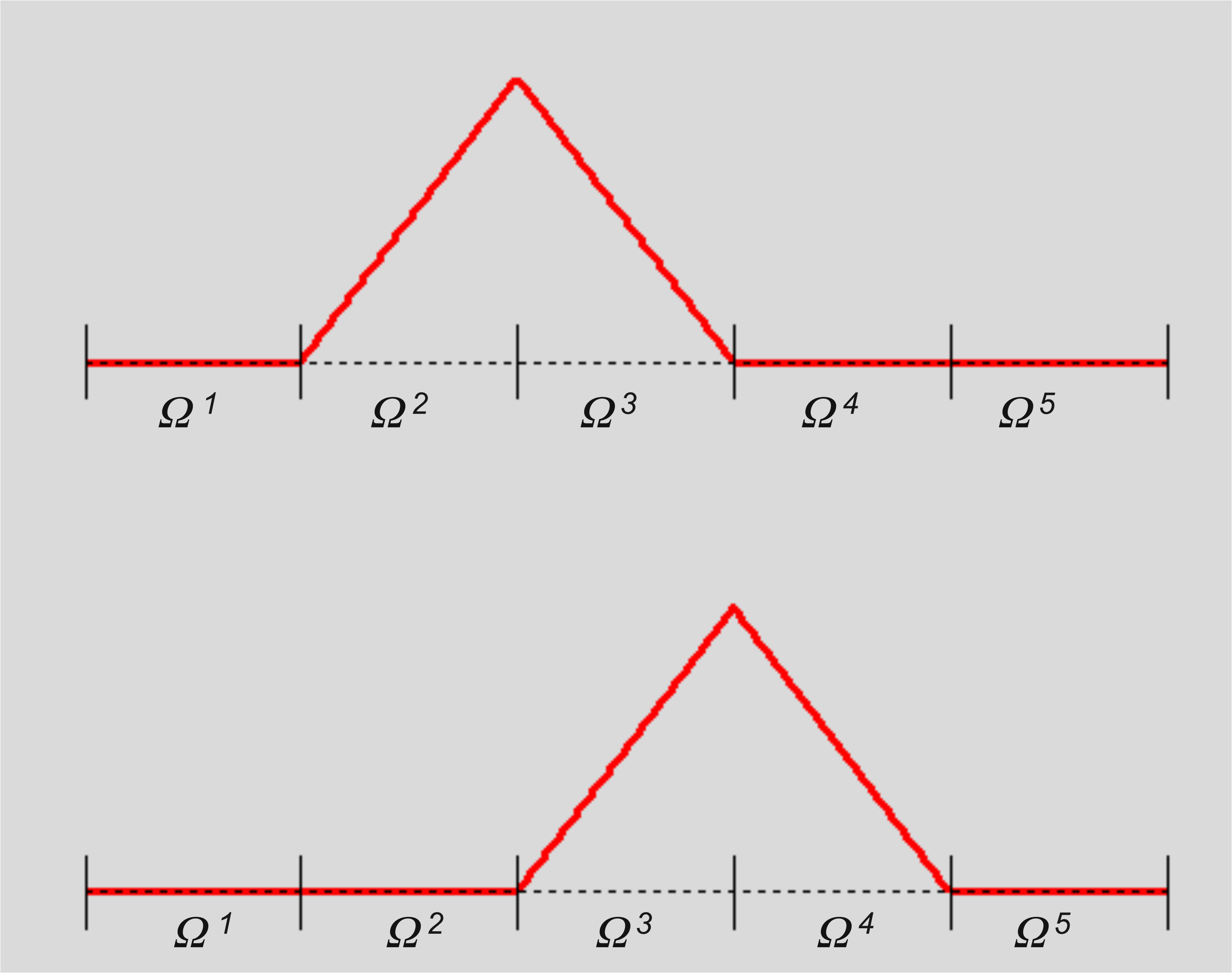

x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Ω^1 Ω^2 Ω^3 Ω^4 Ω^5 Ω^6 Ω^7 Ω^8
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Ω^1 Ω^2 Ω^3 Ω^4

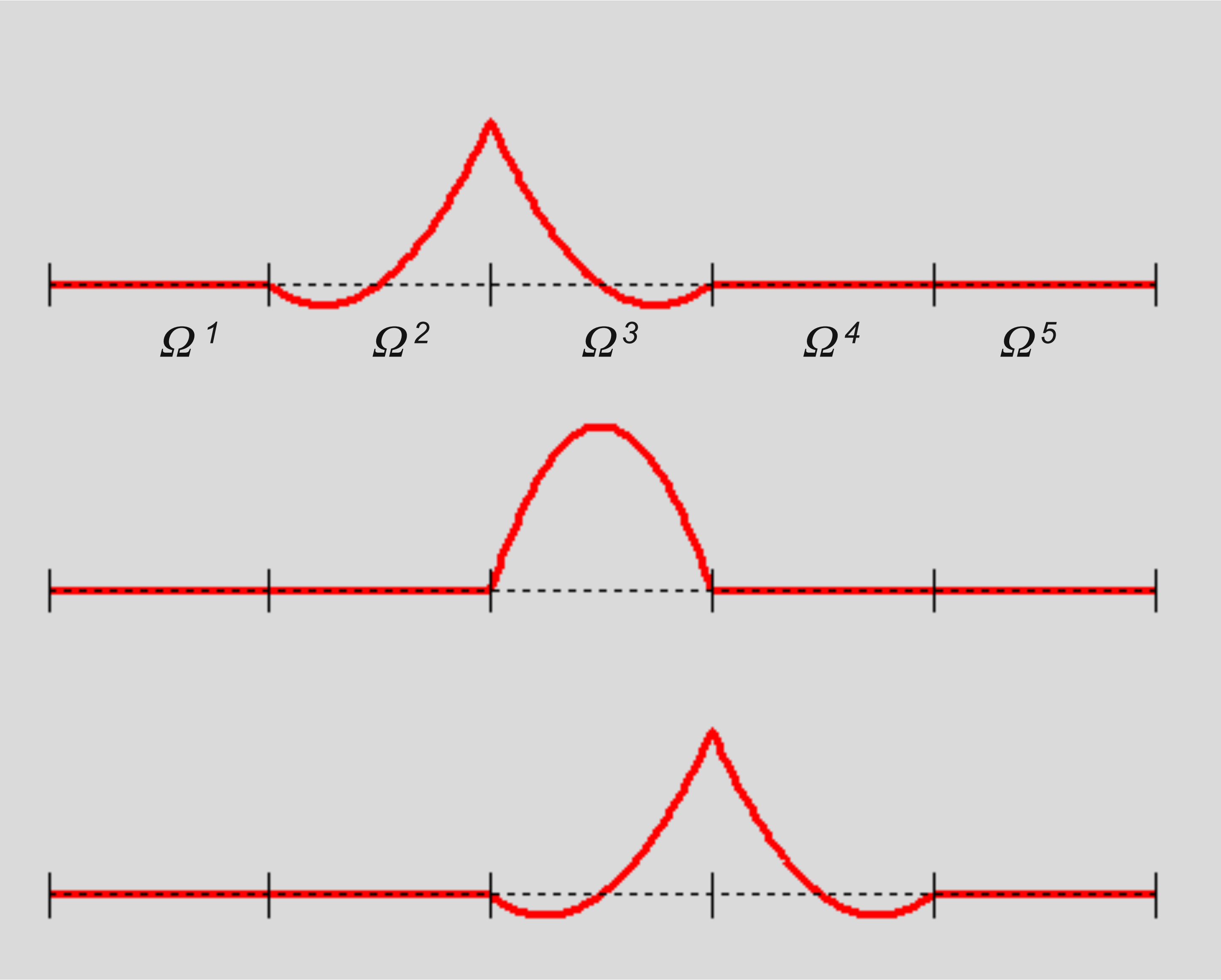
Figure 2: Examples of one-dimensional piecewise linear (left) and piecewise quadratic (right) Lagrangian basis functions, $\phi_2(x)$ and $\phi_3(x)$, with associated element support, Ω^e, $e = 1, \ldots, E$.

- Linear case results in A being tridiagonal (b.w. = 1)
- Q: What is matrix bandwidth for piecewise quadratic case?

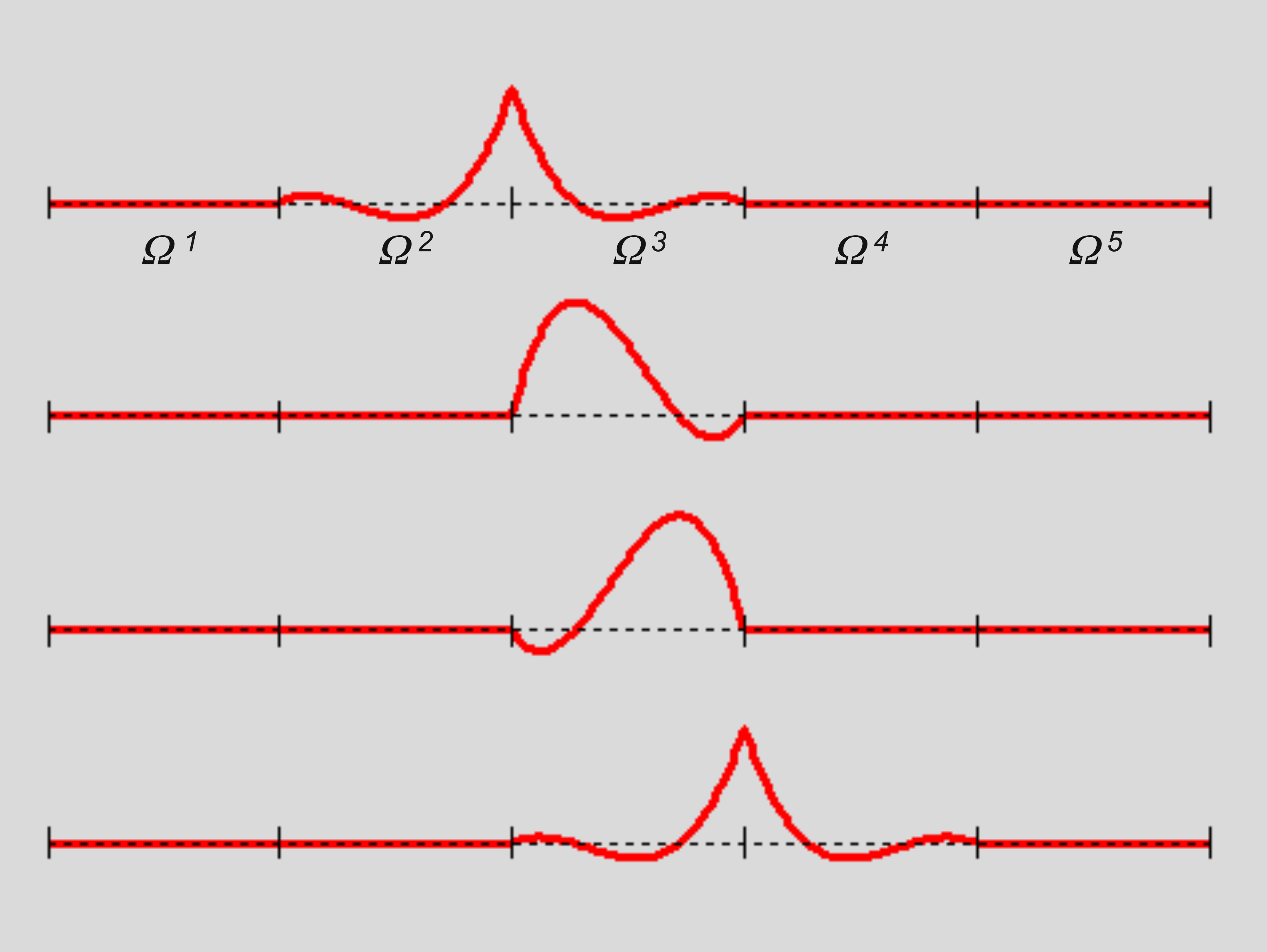
Basis functions for N=1, E=5 on element 3.



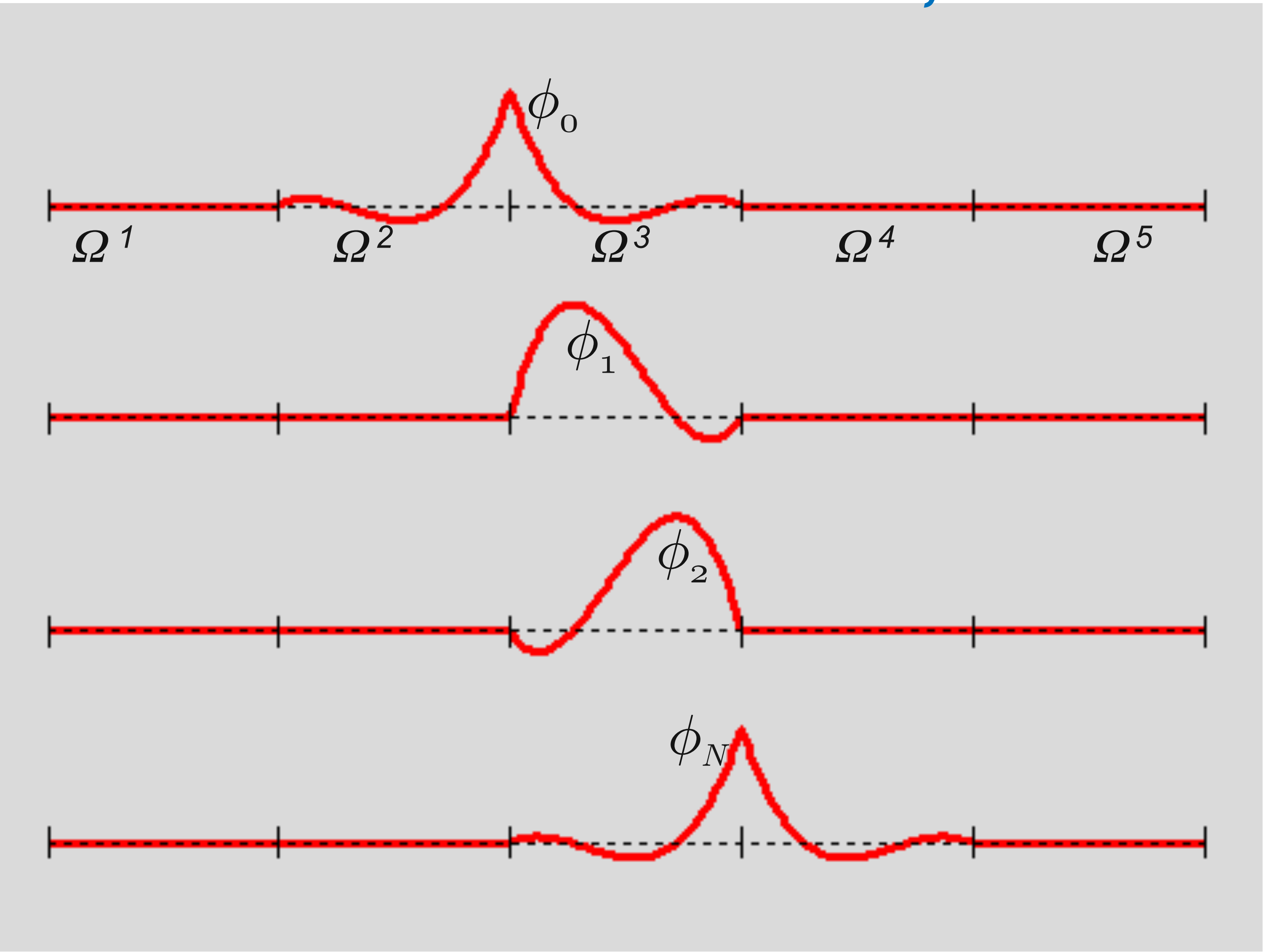
Ω^3 basis functions for N=2, E=5



Ω^3 basis functions for N=3, E=5

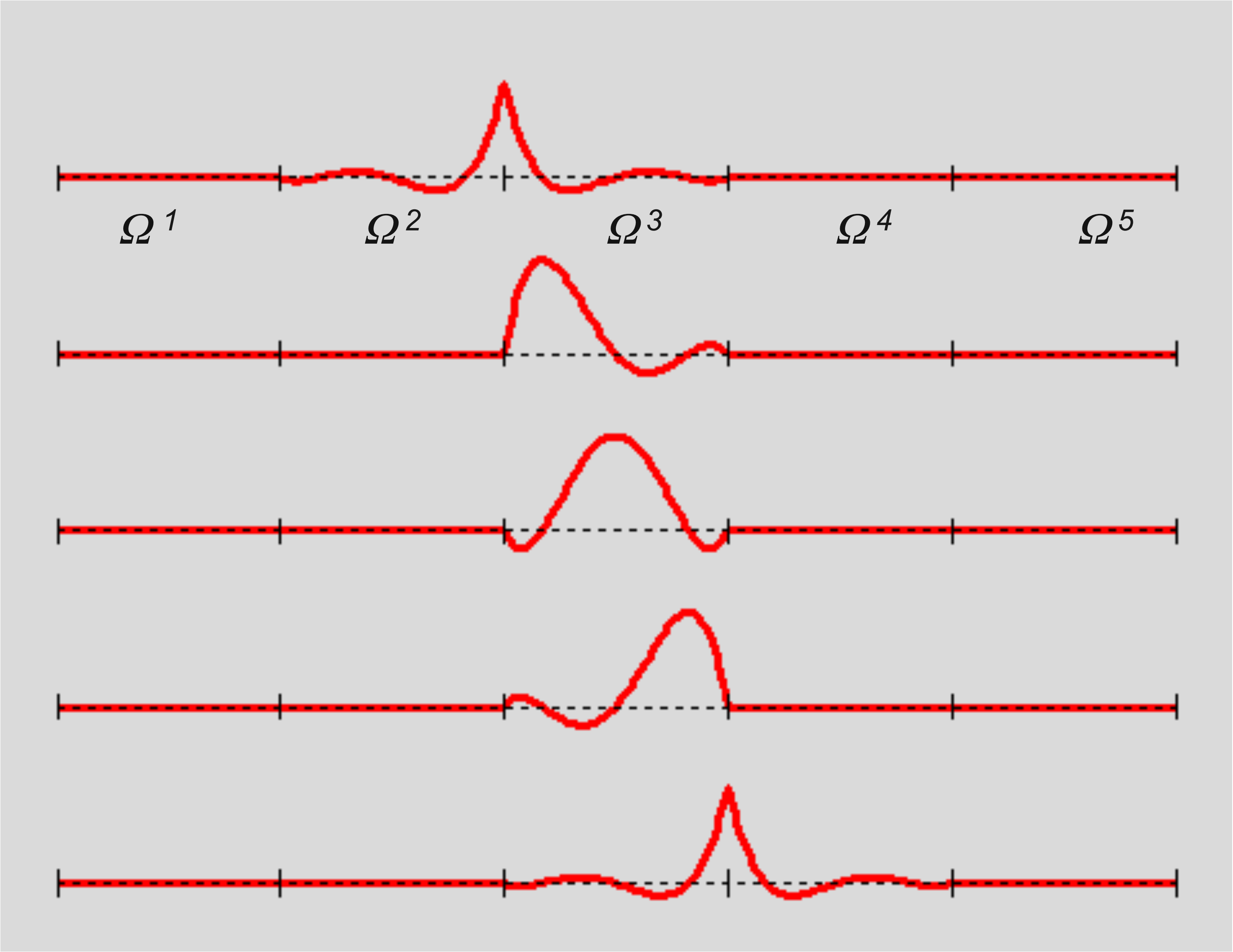


Ω^3 basis functions for N=3, E=5

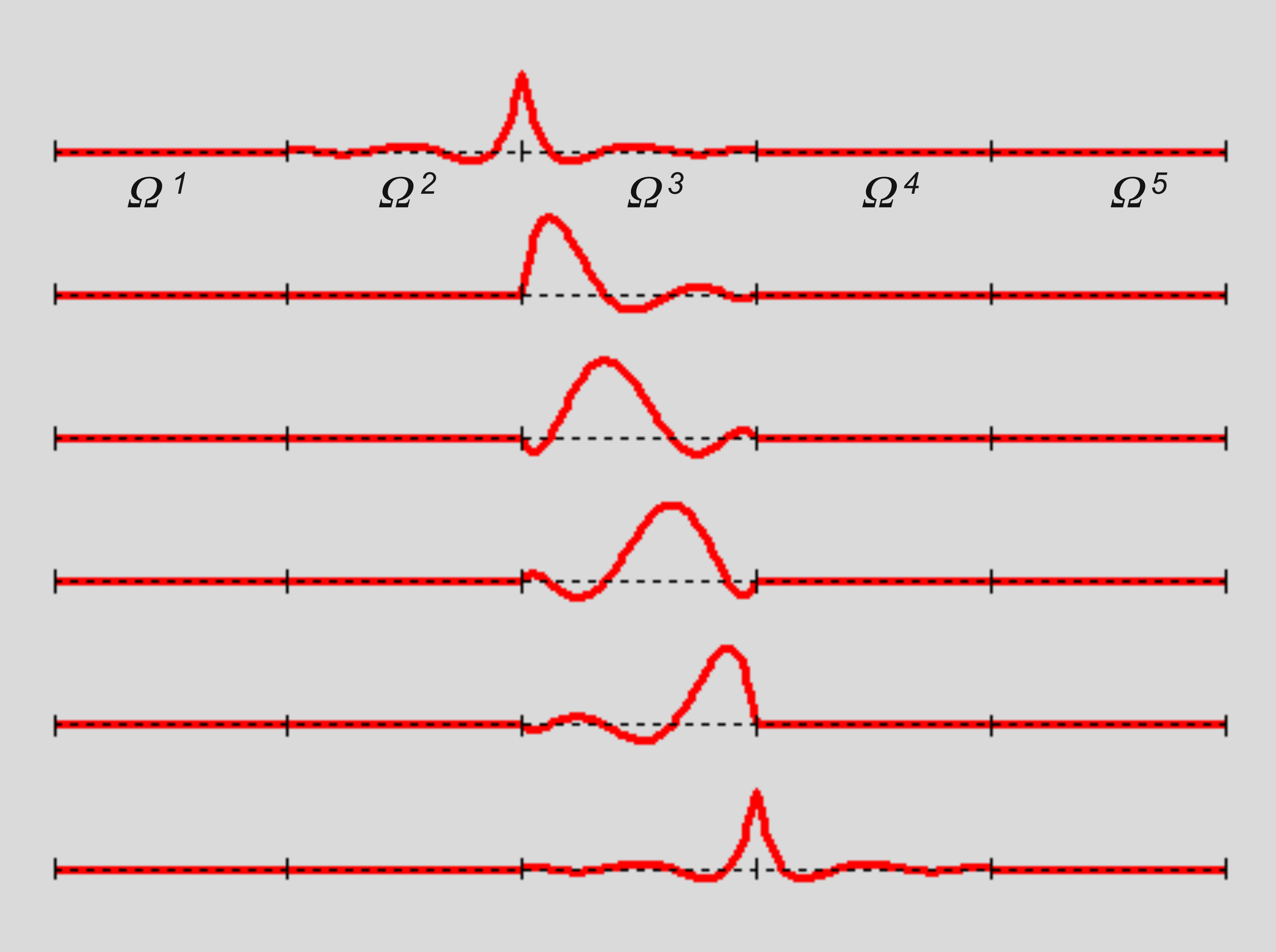


- Notice that ϕ_0 and ϕ_N are also nonzero in the neighboring elements, because of the requirement $X^N \subset H^1$.

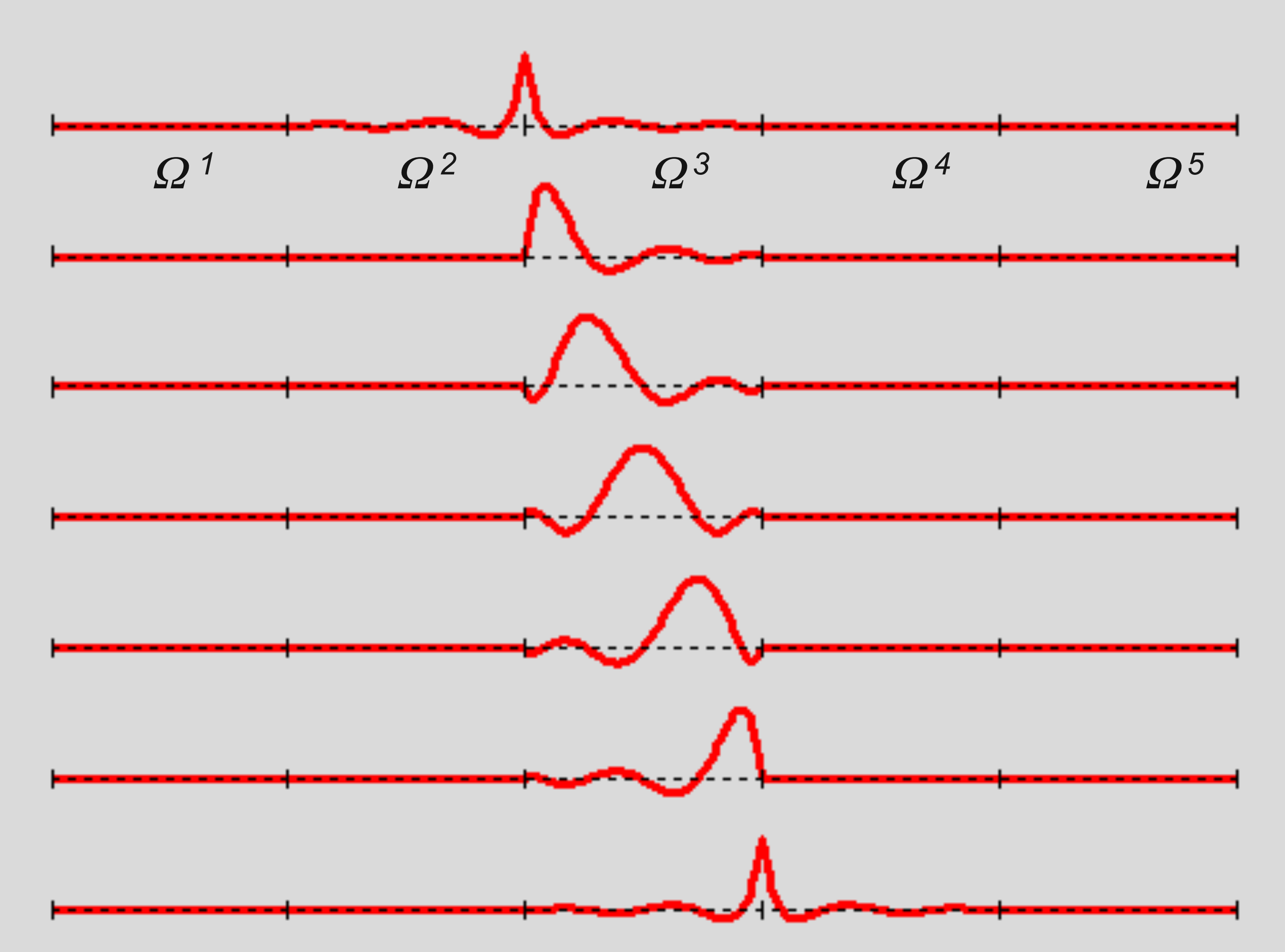
Ω^3 basis functions for N=4, E=5



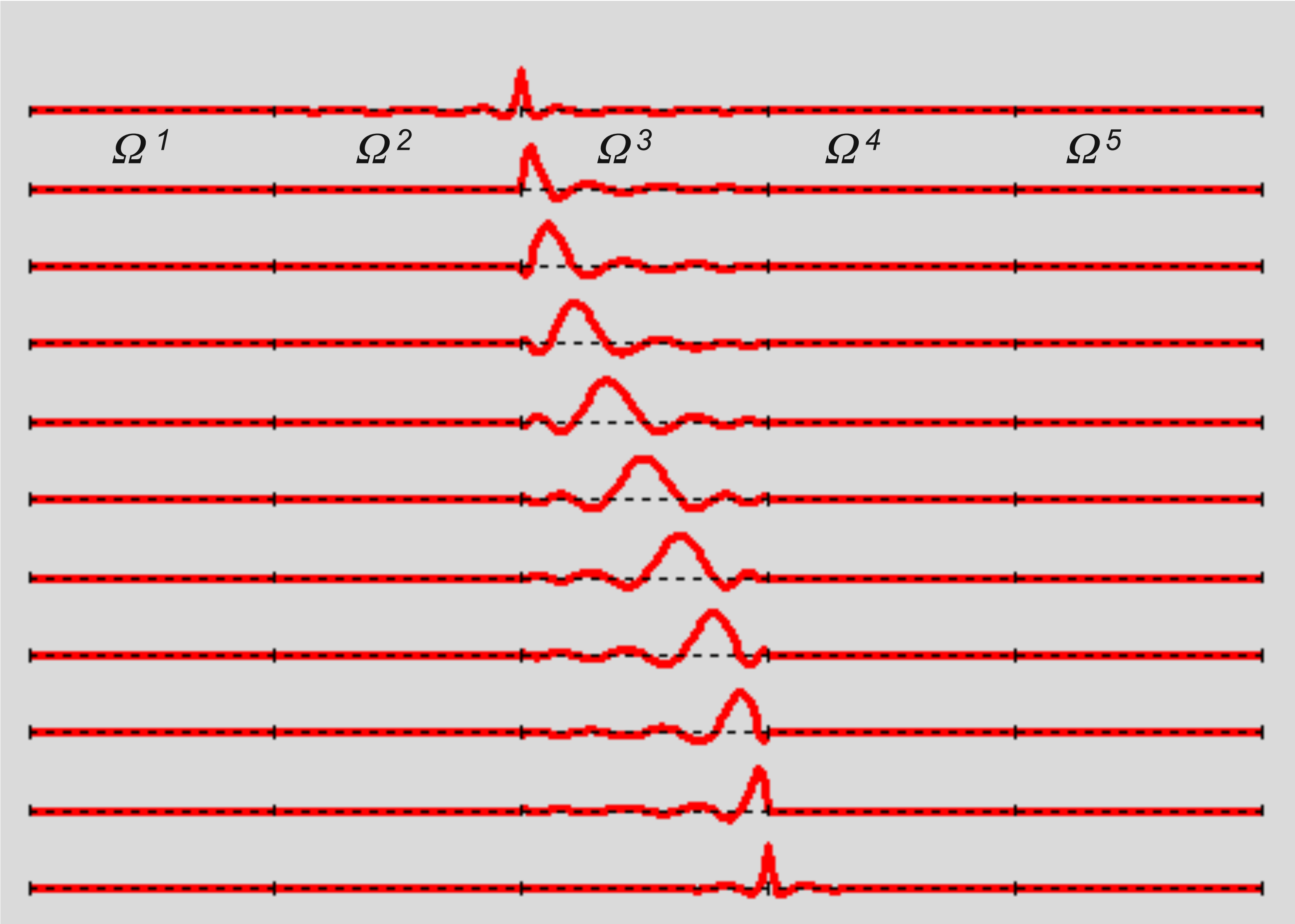
Ω^3 basis functions for N=5, E=5



Ω^3 basis functions for N=6, E=5



Ω^3 basis functions for N=10, E=5

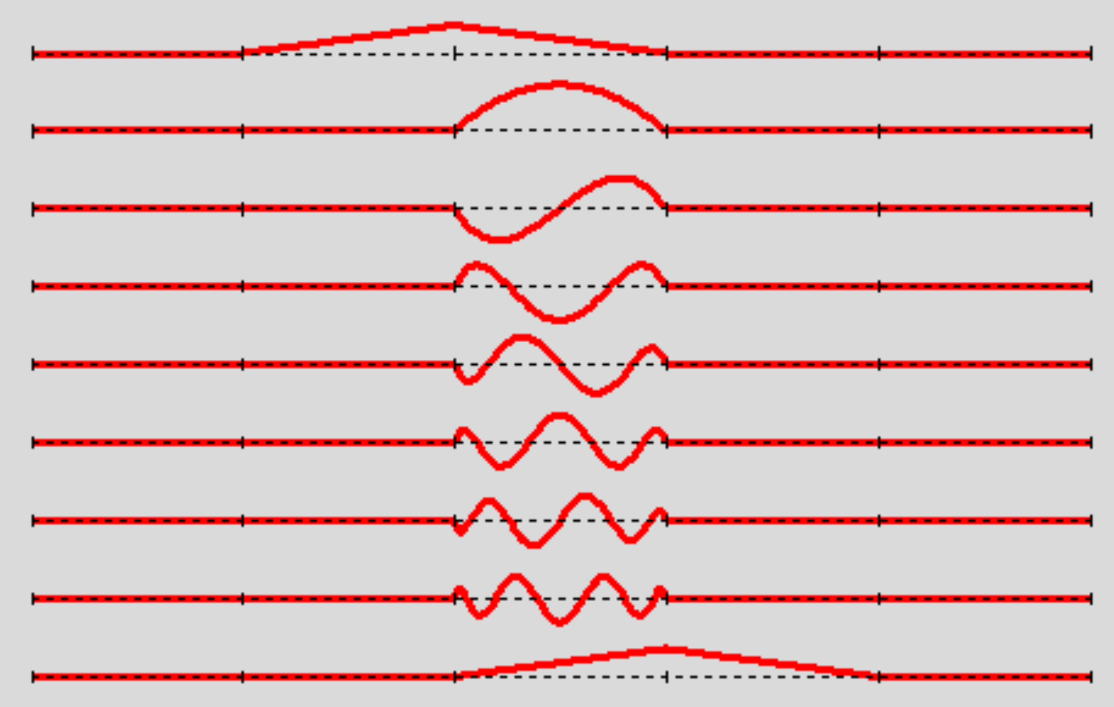


Local Modal Bases, N=8

For $k = 0$ or N, boundary modes.

For $k = 1, \ldots, N-1$,

$$\phi_k(\xi) \;=\; L_{k+1}(\xi) \;-\; L_{k-1}(\xi)$$



•Modal bases are particularly useful for *filtering* (higher k → higher frequency).

•It is easy to convert between *stable* nodal and modal bases.

Working with 1D Nodal Bases on GLL Points

Quadrature: Trapezoidal Rule

Let $\mathcal{I} \;:=\; \int_a^b f(x)\,dx \;\approx\; \sum_{j=0}^{N} w_j f(x_j) \;=:\; Q_N.$

For trapezoidal rule (with uniform spacing, say),

$$x_j \;=\; a \,+\, j\cdot\Delta x, \;\; \Delta x \;:=\; (b-a)/N,$$

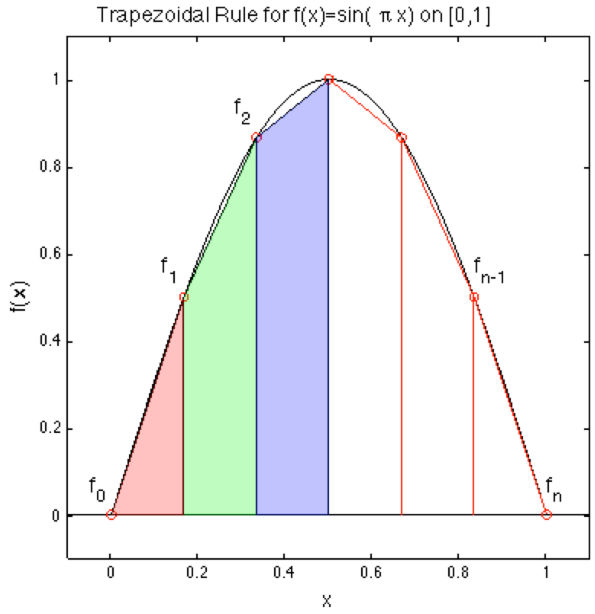
$$w_j \;=\; \Delta x, \; j \;=\; 1,\, 2,\, \ldots, n-1$$

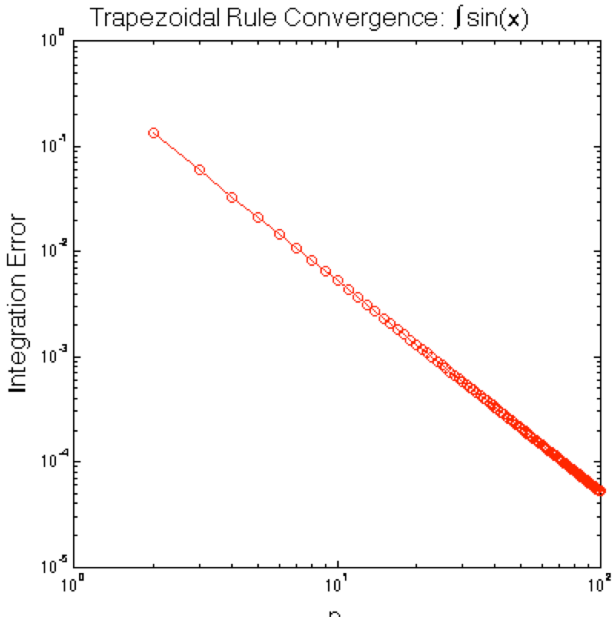
$$w_0 \;=\; w_n \;=\; \frac{1}{2}\Delta x.$$

- Convergence is $O(N^{-2})$:

$$|\,\mathcal{I} \,-\, Q_N\,| \;\sim\; CN^{-2}$$

trap_v_gll.m, trap_txt.m





Working with 1D Nodal Bases on GLL Points

Gauss-Lobatto-Legendre Quadrature

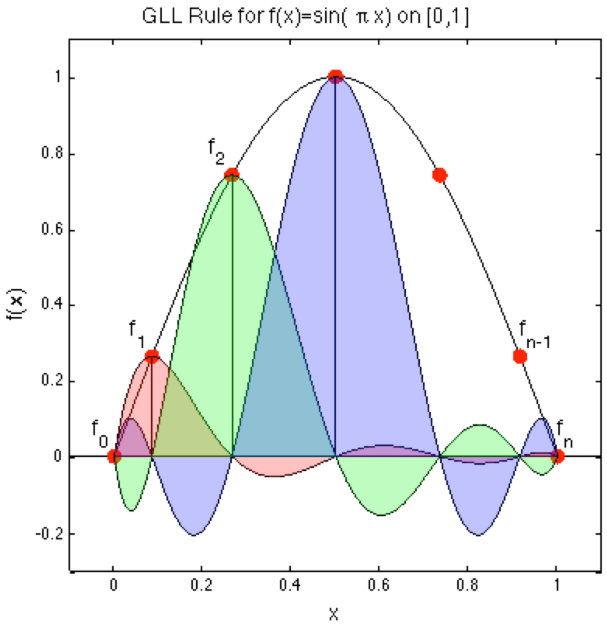
Let $\mathcal{I} \;:=\; \int_a^b f(x)\,dx \;\approx\; \sum_{j=0}^{N} w_j f(x_j) \;=:\; Q_N.$

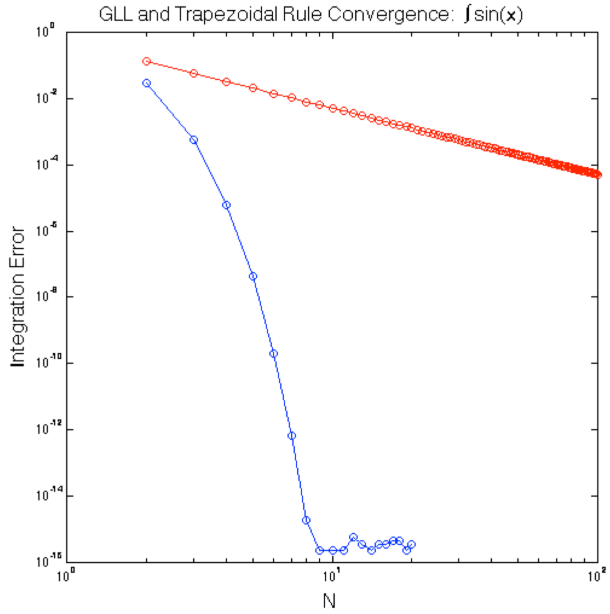
$$
\begin{aligned}
x_j &= a + \frac{b-a}{2}(\xi_j + 1) \\
\xi_j &= \text{GLL quadrature points} \;=\; \text{zeros of}\,(1-\xi^2)\,L_N'(\xi) \\
w_j &= \frac{b-a}{2}\int_{-1}^{1} h_j(\xi)\,d\xi \\
&= \frac{b-a}{2}\rho_j, \quad \rho_j \;:=\; \text{GLL quadrature weight on } [-1,1]
\end{aligned}
$$

- For smooth functions, convergence is $O(e^{-\sigma N})$:

$$|\mathcal{I} - Q_N| \;\sim\; Ce^{-\sigma N}, \quad \sigma > 0.$$

trap_v_gll.m, gll_txt.m





Working with 1D Nodal Bases on GLL Points

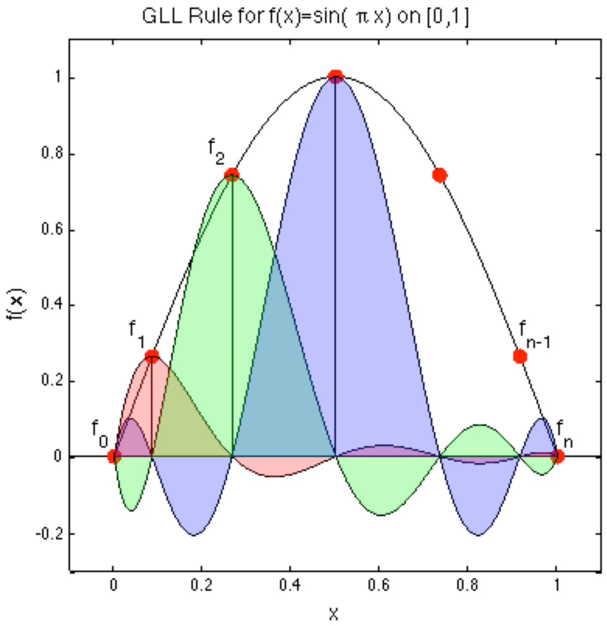
Gauss-Lobatto-Legendre Quadrature

Let $\mathcal{I} \;:=\; \int_a^b f(x)\,dx \;\approx\; \sum_{j=0}^{N} w_j f(x_j) \;=:\; Q_N.$

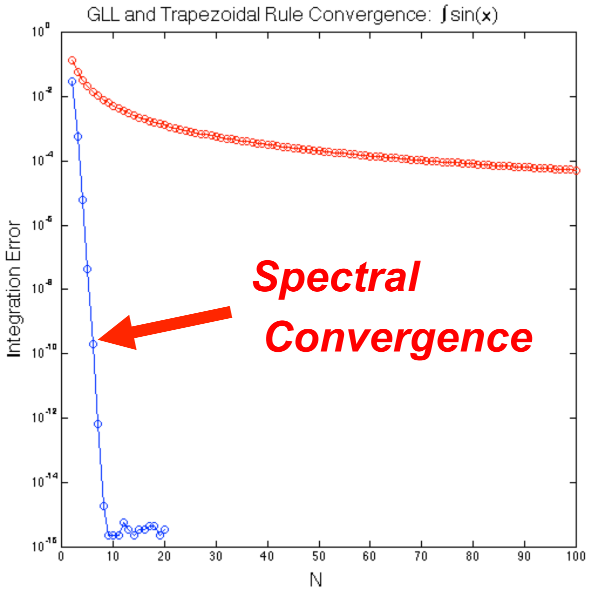
$$
\begin{aligned}
x_j &= a + \frac{b-a}{2}(\xi_j + 1) \\
\xi_j &= \text{GLL quadrature points} = \text{zeros of } (1-\xi^2)\,L_N'(\xi) \\
w_j &= \frac{b-a}{2}\int_{-1}^{1} h_j(\xi)\,d\xi \\
&= \frac{b-a}{2}\rho_j, \quad \rho_j := \text{GLL quadrature weight on } [-1,1]
\end{aligned}
$$

- For smooth functions, convergence is $O(e^{-\sigma N})$:

$$|\mathcal{I} - Q_N| \;\sim\; Ce^{-\sigma N}, \quad \sigma > 0.$$



GLL Rule for f(x)=sin(π x) on [0,1]

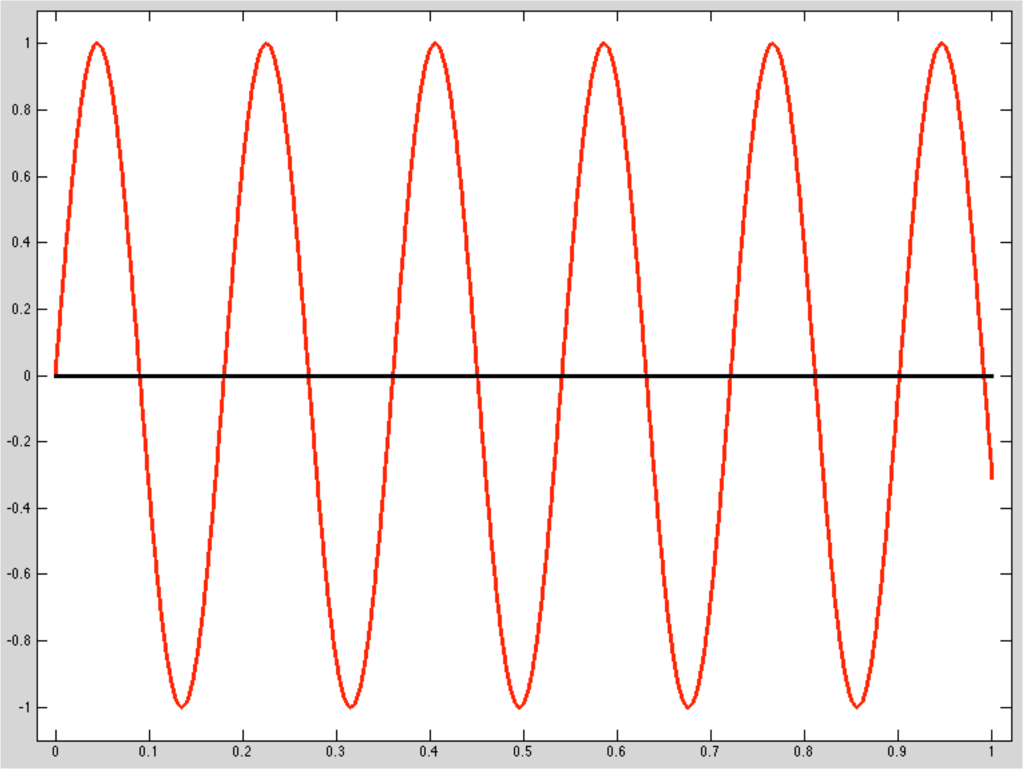


GLL and Trapezoidal Rule Convergence: ∫sin(x)

Spectral Convergence

Working with 1D Nodal Bases

- What is the convergence behavior for highly oscillatory functions?



trap_v_gll_k.m

Working with SEM Bases – 1D

- Keys to high-performance in 3D:
  1. Low numerical dispersion (2x savings in *each* direction, 8x overall)
  2. Element-by-element assembly of solution and data
  3. Use of GLL-based Lagrangian interpolants and quadrature
     - *diagonal mass matrix, fast operator evaluation*
  4. Global (*and local!)* matrix-free operator evaluation
  5. Fast tensor-product based local operator evaluation
  6. Fast tensor-product based local inverses
  7. Matrix-matrix product based kernels

- Only 1—3 are applicable in 1D.
- We'll start with 2 and 3, and come back to 4-7 shortly.

Working with SEM Bases – 1D

- Recall our global system to be solved:

$$A\,\underline{u} = B\,\underline{f}$$

- For most discretizations (finite difference, finite volume, finite element, spectral element, etc.) *iterative solvers* are the fastest possible in 3D.

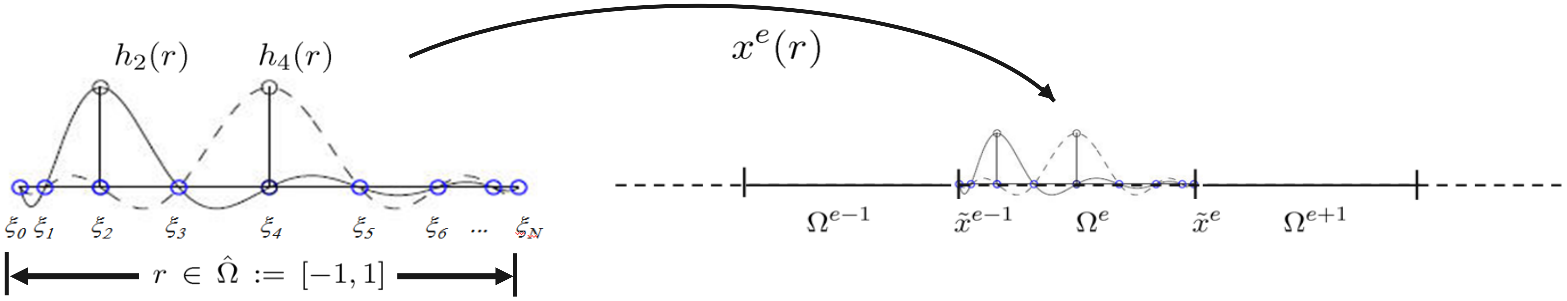
- These solvers require only the *action* of a matrix times a vector (usually implemented via a subroutine) and do *not* require explicit formation of the matrix or its *LU* factorization.

- Thus, we consider *matrix-free* operator evaluation in which we never form the global nor (ultimately in 2D or 3D) the local stiffness matrix.

- It is nonetheless useful to understand the matrix assembly process, as notation and analysis in linear algebra is quite helpful.
  - Also, for matlab, it generally pays to assemble the 1D matrices.

Spectral Element Bases, 1D

- We transform coordinates from $\hat{\Omega}$ to Ω^e via affine mappings, $x^e(r)$.



$h_2(r)$
$h_4(r)$
$x^e(r)$
ξ_0 ξ_1 ξ_2 ξ_3 ξ_4 ξ_5 ξ_6 ... ξ_N
$r \in \hat{\Omega} := [-1,1]$
Ω^{e-1}
$\tilde{x}^{e-1}$
Ω^e
$\tilde{x}^e$
Ω^{e+1}

- On Ω^e we have,

$$u(x)|_{\Omega^e} \quad = \quad \sum_{j=0}^{N} u_j^e \, h_j(r) \qquad r \in \hat{\Omega} := [-1,1]$$

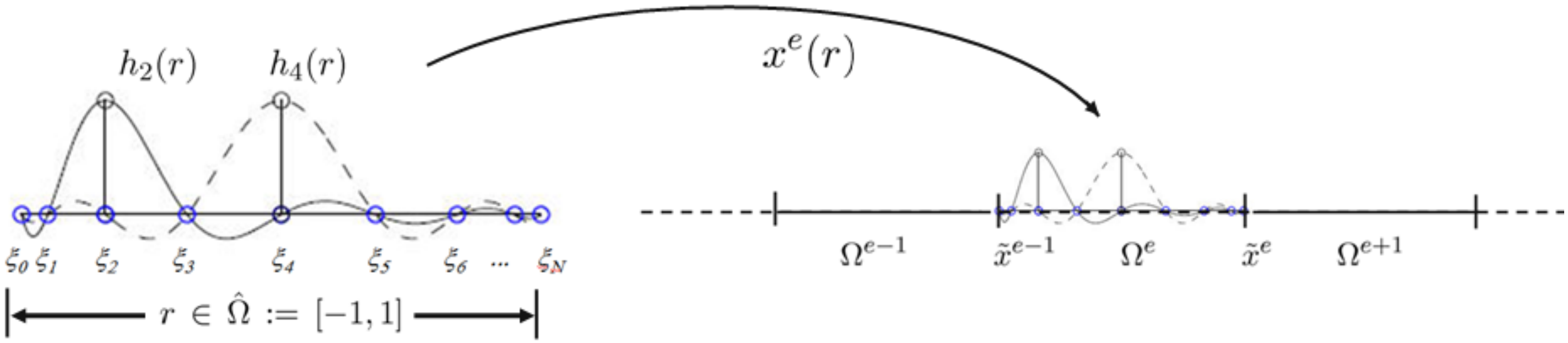
$$x^e(r) \quad := \quad x|_{\Omega^e} = \tilde{x}^{e-1} + \frac{\tilde{x}^e - \tilde{x}^{e-1}}{2}(r+1)$$

$$h_j(r) \quad \in \quad \mathbb{P}_N(r)$$

$$h_j(\xi_i) \quad = \quad \delta_{ij}, \qquad i,j \in [0,\ldots,N]^2$$

$$\xi_i \quad = \quad \text{GLL quadrature points} \in [-1,1]$$

Spectral Element Bases, 1D


$h_2(r)$
$h_4(r)$
$x^e(r)$
ξ_0 ξ_1 ξ_2 ξ_3 ξ_4 ξ_5 ξ_6 ... ξ_N
$r \in \hat{\Omega} := [-1,1]$
Ω^{e-1}
$\tilde{x}^{e-1}$
Ω^e
$\tilde{x}^e$
Ω^{e+1}

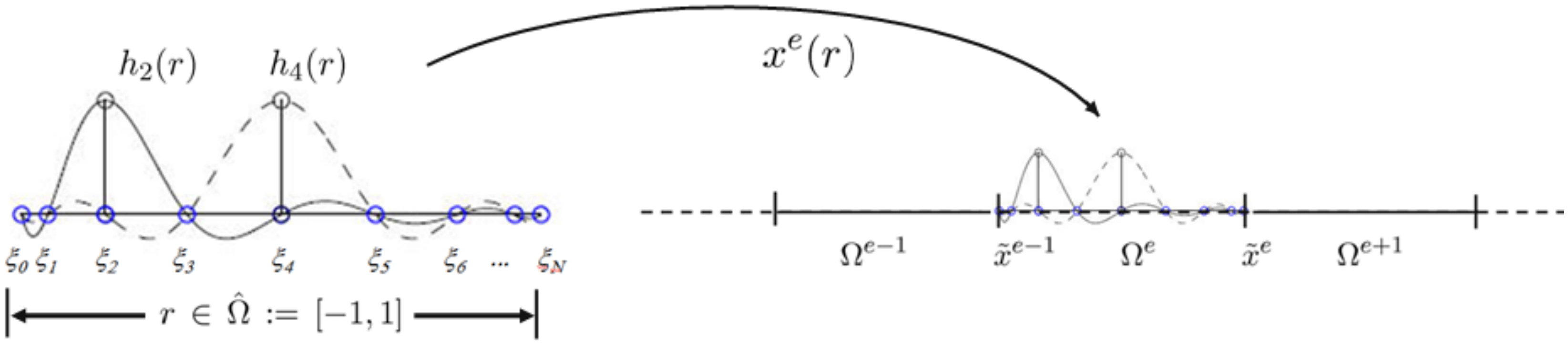
Return to the WRT and consider $v,\ u \in X^N$ (but not X_0^N for now). Let

$$\mathcal{I} \quad := \quad \int_\Omega \frac{dv}{dx}\frac{du}{dx}\,dx \;=\; \sum_{e=1}^{E} \int_{\Omega^e} \frac{dv}{dx}\frac{du}{dx}\,dx. \;=\; \sum_{e=1}^{E} \mathcal{I}^e,$$

With $L^e := \tilde{x}^e - \tilde{x}^{e-1}$ we have,

$$\mathcal{I}^e \quad := \quad \int_{\Omega^e} \frac{dv}{dx}\frac{du}{dx}\,dx \;=\; \frac{2}{L^e}\int_{-1}^{1} \frac{dv}{dr}\frac{du}{dr}\,dr.$$

Spectral Element Bases, 1D


$h_2(r)$
$h_4(r)$
$x^e(r)$
ξ_0 ξ_1 ξ_2 ξ_3 ξ_4 ξ_5 ξ_6 ... ξ_N
$r \in \hat{\Omega} := [-1,1]$
Ω^{e-1}
$\tilde{x}^{e-1}$
Ω^e
$\tilde{x}^e$
Ω^{e+1}

Using

$$u|_{\Omega^e} = u^e(r) := \sum_{j=0}^{N} u_j^e \, h_j(r), \qquad v|_{\Omega^e} = v^e(r) := \sum_{i=0}^{N} v_i^e \, h_i(r),$$

we can readily compute the derivatives,

$$\frac{du^e}{dr} = \sum_{j=0}^{N} u_j^e \frac{dh_j}{dr}, \qquad \frac{dv^e}{dr} = \sum_{i=0}^{N} v_i^e \frac{dh_i}{dr}.$$

Local 1D Stiffness Matrix

Inserting the local basis into the local integral yields

$$\begin{aligned}
\mathcal{I}^e \quad &:= \quad \int_{\Omega^e} \frac{dv}{dx}\frac{du}{dx}\,dx \;=\; \frac{2}{L^e}\int_{-1}^{1}\left(\sum_{i=0}^{N}\frac{dh_i}{dr}\,v_i^e\right)\left(\sum_{j=0}^{N}\frac{dh_j}{dr}\,u_j^e\right)dr \\
&:= \quad \sum_{i=0}^{N} v_i^e \left(\frac{2}{L^e}\int_{-1}^{1}\frac{dh_i}{dr}\frac{dh_j}{dr}\,dr\right)u_j^e \\
&:= \quad \sum_{i=0}^{N} v_i^e\, A_{ij}^e u_j^e,
\end{aligned}$$

where

$$A_{ij}^e \;:=\; \frac{2}{L^e}\hat{A}_{ij}, \qquad \text{and} \qquad \hat{A}_{ij} \;:=\; \int_{-1}^{1}\frac{dh_i}{dr}\frac{dh_j}{dr}\,dr.$$

Assembly of 1D Stiffness Matrix

If we define $\underline{u}^e := (u_0^e \, u_1^e \, \ldots \, u_N^e)^T$ and similarly for $\underline{v}^e$, we have

$$\mathcal{I}^e \quad = \quad \sum_{i=0}^{N} \sum_{j=0}^{N} v_i^e \, A_{ij}^e u_j^e \; = \; (\underline{v}^e)^T \, A^e \underline{u}^e.$$

Let

$$\underline{u}_L := \begin{pmatrix} \underline{u}^1 \\ \underline{u}^2 \\ \vdots \\ \underline{u}^e \\ \vdots \\ \underline{u}^E \end{pmatrix}, \quad A_L := \begin{pmatrix} A^1 & & & & & \\ & A^2 & & & & \\ & & \ddots & & & \\ & & & A^e & & \\ & & & & \ddots & \\ & & & & & A^E \end{pmatrix}$$

Define $\underline{v}_L$ similarly using $\underline{v}^e$.

Assembly of 1D Stiffness Matrix

The left-hand side of our WR statement reads

$$\mathcal{I} = \sum_{e=1}^{E} \mathcal{I}^e = \sum_{e=1}^{E} \boxed{(\underline{v}^e)^T \, A^e \underline{u}^e}$$

$$= \begin{pmatrix} \underline{v}^1 \\ \underline{v}^2 \\ \vdots \\ \underline{v}^e \\ \vdots \\ \underline{v}^E \end{pmatrix}^T \begin{pmatrix} A^1 & & & & & \\ & A^2 & & & & \\ & & \ddots & & & \\ & & & A^e & & \\ & & & & \ddots & \\ & & & & & A^E \end{pmatrix} \begin{pmatrix} \underline{u}^1 \\ \underline{u}^2 \\ \vdots \\ \underline{u}^e \\ \vdots \\ \underline{u}^E \end{pmatrix},$$

$$= \underline{v}_L^T A_L \, \underline{u}_L .$$

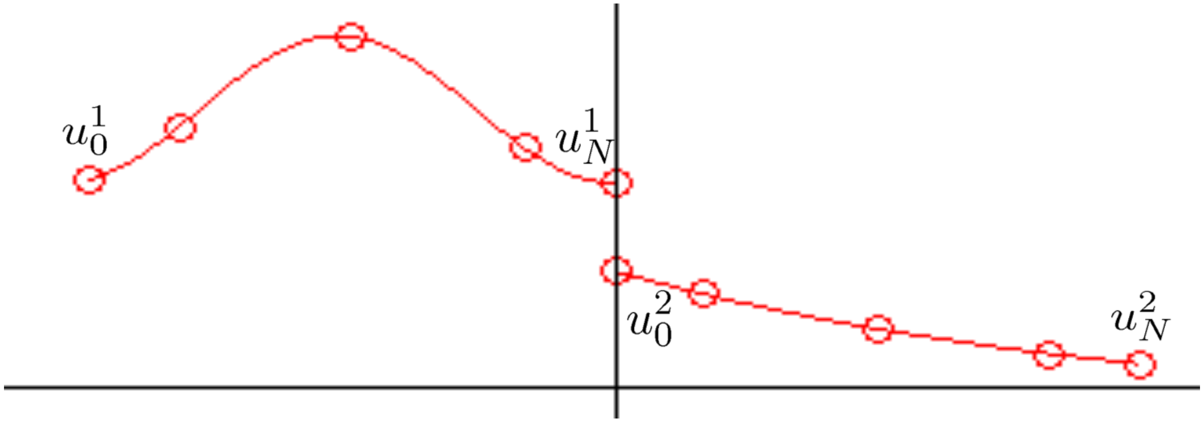
- This is an important picture, so let's look at it a bit more deeply.

Working with the unassembled matrix, A_L

$$\mathcal{I} = \begin{pmatrix} \underline{v}^1 \\ \underline{v}^2 \\ \vdots \\ \underline{v}^e \\ \vdots \\ \underline{v}^E \end{pmatrix}^T \underbrace{\begin{pmatrix} A^1 & & & & & \\ & A^2 & & & & \\ & & \ddots & & & \\ & & & A^e & & \\ & & & & \ddots & \\ & & & & & A^E \end{pmatrix}}_{A_L} \begin{pmatrix} \underline{u}^1 \\ \underline{u}^2 \\ \vdots \\ \underline{u}^e \\ \vdots \\ \underline{u}^E \end{pmatrix} = \underline{v}_L^T A_L \, \underline{u}_L .$$

- A_L has *precisely* the same structure in higher space dimensions.
- In 2D and 3D problems, we work exclusively with A^e and $\underline{u}^e$, $e = 1, \dots, E$.
- In fact, we never even form A^e, but just compute the *action* of A^e on $\underline{u}^e$.
- The *physics* of the operator is embedded in the A^es.
- It is clear that $A^e \underline{u}^e$, $e = 1, \dots, E$ can be computed independently, *in parallel!*
- Keep in mind that $\underline{u}^e$ is simply the set of *local* basis coefficients on Ω^e.

What about Continuity ?



Because $X^N \subset \mathcal{H}^1$, we must have u (and v) be continuous.
Thus, we can't allow functions like the one above.

If $u \in X^N$ then we must have $u_N^1 \equiv u_0^2$ or, in general, $u_N^e \equiv u_0^{e+1}$.

In higher space dimensions, a similar rule applies.
For example, in 2D, we have for any $u \in X^N$

$$u_{ij}^e \equiv u_{\hat{\imath}\hat{\jmath}}^{\hat{e}} \qquad \textit{iff} \qquad \mathbf{x}_{ij}^e \equiv \mathbf{x}_{\hat{\imath}\hat{\jmath}}^{\hat{e}},$$

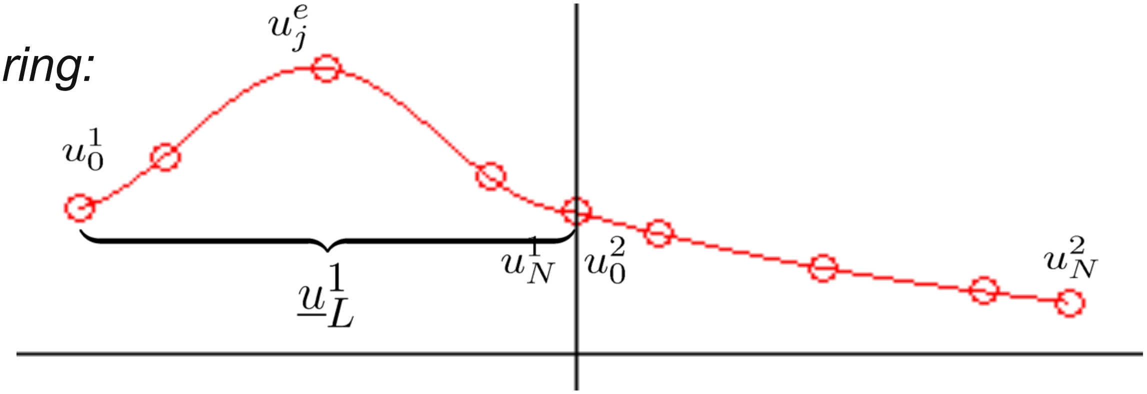
where $\mathbf{x}_{ij}^e = (x_{ij}^e,\, y_{ij}^e)$.

Continuity is reflected by global numbering:

For a continuous $u(x)$ (i.e., $u \in X^N$), we have:

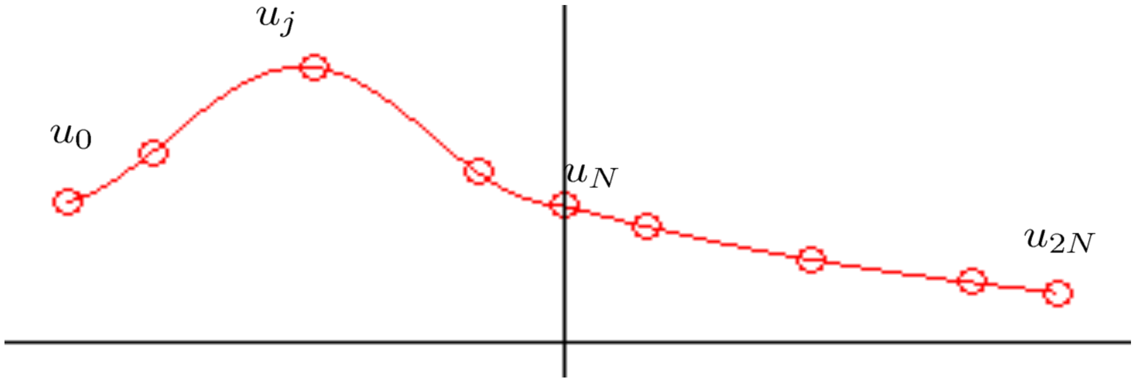
Local (elemental) numbering:

$$\underline{u}_L := \begin{pmatrix} \underline{u}^1 \\ \underline{u}^2 \\ \vdots \\ \underline{u}^e \\ \vdots \\ \underline{u}^E \end{pmatrix},$$



Global numbering:

$$\underline{\bar{u}} := \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_j \\ \vdots \\ u_{n+1} \end{pmatrix},$$



Recall, for $u \in X_0^N$ we reqiure $u(0) = u(1) = 0$, which implies $u_0 = u_{n+1} = 0$ in the global notation, so we will have only $\underline{u} = (u_1\, u_2\, \dots\, u_n)^T$ as degrees-of-freedom.

Continuity is reflected by global numbering:

For *any* $u \in X^N$, there is a Boolean (1s and 0s) matrix Q such that $\underline{u}_L = Q\underline{\bar{u}}$, where $\underline{u}_L$ is the *local* nodal representation and $\underline{\bar{u}}$ is the global representation, including boundary values.

For example, in the preceding case, we have:

$$\begin{pmatrix} u_0^1 \\ u_1^1 \\ u_2^1 \\ u_3^1 \\ u_4^1 \\ u_0^2 \\ u_1^2 \\ u_2^2 \\ u_3^2 \\ u_4^2 \end{pmatrix} = \begin{pmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix}$$

$$\underline{u}_L \quad = \qquad\qquad Q \qquad\qquad \underline{\bar{u}}$$

Note that $u_4^1 = u_0^2$ are copies of u_4 by virtue of the pair of 1s in column 5.

Continuity is reflected by global numbering:

One the left, we have the distribution of the local basis coefficients given in $\underline{u}_L = (\underline{u}^1 \; \underline{u}^2)^T$:

$$\begin{array}{c} \\ \\ \underline{u}^1 \\ \\ \\ \\ \underline{u}^2 \\ \\ \\ \\ \end{array}
\begin{pmatrix} u_0^1 \\ u_1^1 \\ u_2^1 \\ u_3^1 \\ u_4^1 \\ u_0^2 \\ u_1^2 \\ u_2^2 \\ u_3^2 \\ u_4^2 \end{pmatrix}
=
\begin{pmatrix}
1 & & & & & & & & \\
 & 1 & & & & & & & \\
 & & 1 & & & & & & \\
 & & & 1 & & & & & \\
 & & & & 1 & & & & \\
 & & & & 1 & & & & \\
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1
\end{pmatrix}
\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix}$$

$$\underline{u}_L \quad = \quad Q \quad \bar{\underline{u}}$$

Note that $u_4^1 = u_0^2$ are copies of u_4 by virtue of the pair of 1s in column 5.

Continuity is reflected by global numbering:

Note that $u_4^1 = u_0^2$ are copies of u_4 by virtue of the pair of 1s in column 5.

$$
\begin{pmatrix} u_0^1 \\ u_1^1 \\ u_2^1 \\ u_3^1 \\ u_4^1 \\ u_0^2 \\ u_1^2 \\ u_2^2 \\ u_3^2 \\ u_4^2 \end{pmatrix}
=
\begin{pmatrix}
1 & & & & & & & & \\
& 1 & & & & & & & \\
& & 1 & & & & & & \\
& & & 1 & & & & & \\
& & & & 1 & & & & \\
& & & & 1 & & & & \\
& & & & & 1 & & & \\
& & & & & & 1 & & \\
& & & & & & & 1 & \\
& & & & & & & & 1
\end{pmatrix}
\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \boxed{u_4} \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix}
$$

$$\underline{u}_L = Q \; \underline{\bar{u}}$$

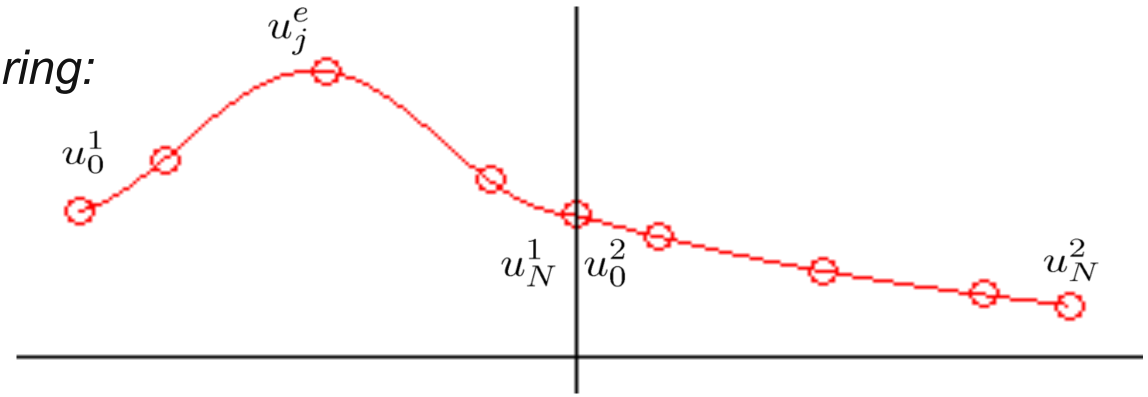
Q: What is the result if we compute some vector $\underline{\tilde{w}} = Q^T \underline{w}_L$?

Continuity is reflected by global numbering:

For a continuous $u(x)$ (i.e., $u \in X^N$), we have:

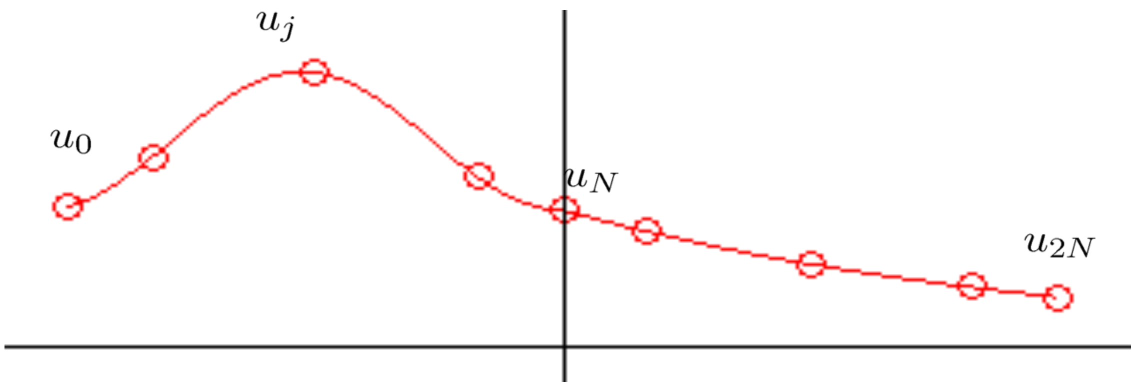
Local (elemental) numbering:

$$\underline{u}_L := \begin{pmatrix} \underline{u}^1 \\ \underline{u}^2 \\ \vdots \\ \underline{u}^e \\ \vdots \\ \underline{u}^E \end{pmatrix},$$



Global numbering:

$$\underline{\bar{u}} := \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_j \\ \vdots \\ u_{n+1} \end{pmatrix},$$



Recall, for $u \in X_0^N$ we reqiure $u(0) = u(1) = 0$, which implies $u_0 = u_{n+1} = 0$ in the global notation, so we will have only $\underline{u} = (u_1\, u_2\, \ldots\, u_n)^T$ as degrees-of-freedom.

Continuity is reflected by global numbering:

So, for any $u \in X^N \subset \mathcal{H}^1$, there exists a global vector $\underline{\bar{u}} = (u_0\, u_1\, u_2\, \dots u_{n+1})^T$ and Boolean matrix Q such that the *local basis coefficients* are given by

$$\underline{u}_L \;=\; Q\, \underline{\bar{u}},$$

where

$$\begin{aligned} \underline{u}_L &:= (\underline{u}^1\, \underline{u}^2\, \dots \underline{u}^e \dots\, \underline{u}^E)^T \\ \underline{u}^e &:= (u_0^e\, u_1^e\, \dots u_N^e)^T, \end{aligned}$$

and

$$\underline{\bar{u}} \;:=\; (u_0\, u_1\, \dots u_j \dots\, u_{n+1})^T.$$

These components allow us to cast the global (solvable) system in terms of local (computable) quantities.

- $\underline{u}_L$ is the *only* vector we work with in the SEM for 2D and 3D problems.

Stiffness Matrix Assembly

Returning to our WR statement and using $\underline{u}_L = Q\underline{\bar{u}}$, we have

$$\begin{aligned} \mathcal{I} &= \underline{v}_L^T A_L \, \underline{u}_L \\ &= (Q\,\underline{\bar{v}})^T A_L \, Q\underline{\bar{u}} \\ &= \underline{\bar{v}}^T Q^T A_L \, Q\underline{\bar{u}} \\ &= \underline{\bar{v}}^T \bar{A} \, \underline{\bar{u}}. \end{aligned}$$

Here, $\bar{A}$ is the *assembled Neumann operator*:

$$\bar{A} \quad := \quad Q^T A_L Q.$$

It has a non-trivial null space of dimension 1 because we have yet to apply the boundary conditions. ($\bar{A}$ times the constant vector is zero.)

At this point we need to restrict $\underline{\bar{v}}$ and $\underline{\bar{u}}$ to apply the boundary conditions, i.e., to ensure $u, v \in X_0^N \subset \mathcal{H}_0^1$.

Application of Homogeneous Dirichlet Conditions

- If $u \in X_0^N$, the global (and local) basis coefficients on the boundary are zero:

$$u_0 \;=\; u_0^1 \;=\; 0, \qquad \text{and} \qquad u_{n+1} \;=\; u_N^E \;=\; 0.$$

- In our original definition of A, we had $\mathcal{I} = \underline{v}^T A \underline{u}$, where the index ranged from 1 to n on the global vectors $\underline{v}$ and $\underline{u}$.

- Therefore we construct a restriction matrix R and prolongation matrix R^T that "cuts off" u_0 and u_{n+1}.

- If $\bar{n} := n + 2$ (or, in general, accounts for all points, inc. boundary), then R is essentially the $\bar{n} \times \bar{n}$ identity matrix with rows corresponding to boundary points deleted.

Application of Homogeneous Dirichlet Conditions

- Here is an example of the restriction matrix applied to yield $\underline{u} = R\underline{\bar{u}}$:

$$
\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix}
=
\begin{pmatrix}
0 & 1 & & & & & & & & \\
 & & 1 & & & & & & & \\
 & & & 1 & & & & & & \\
 & & & & 1 & & & & & \\
 & & & & & 1 & & & & \\
 & & & & & & 1 & & & \\
 & & & & & & & 1 & & \\
 & & & & & & & & 1 & 0
\end{pmatrix}
\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix}
$$

$$
\underline{u} \quad = \qquad R \qquad \underline{\bar{u}}
$$

Application of Homogeneous Dirichlet Conditions

- The real strength of R, however, comes in the application of its transpose, which allows us to generate a $\underline{\bar{u}}$ and, ultimately, $\underline{u}_L$ such that $u \in X_0^N$.
- Below, we see that u_0 and u_{n+1} will be zero, which is what we want.

$$
\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix}
=
\begin{pmatrix}
0 & & & & & & & \\
1 & & & & & & & \\
 & 1 & & & & & & \\
 & & 1 & & & & & \\
 & & & 1 & & & & \\
 & & & & 1 & & & \\
 & & & & & 1 & & \\
 & & & & & & 1 & \\
 & & & & & & & 1 \\
 & & & & & & & 0
\end{pmatrix}
\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix}
$$

$$
\underline{\bar{u}} = R^T \underline{u}
$$

Application of Homogeneous Dirichlet Conditions

- Note that we can also generate a mask, $\mathcal{M} := R^T R$ that will map a function from X^N into X_0^N.

- This is the approach used in Nek5000, where functions are always represented by *local* coefficients.

$$\begin{array}{cccc}
\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix} & \longleftarrow &
\begin{pmatrix}
0 &&&&&&&&& \\
& 1 &&&&&&&& \\
&& 1 &&&&&&& \\
&&& 1 &&&&&& \\
&&&& 1 &&&&& \\
&&&&& 1 &&&& \\
&&&&&& 1 &&& \\
&&&&&&& 1 && \\
&&&&&&&& 1 & \\
&&&&&&&&& 0
\end{pmatrix} &
\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix} \\
\underline{\bar{u}} & \longleftarrow & R^T R & \underline{\bar{u}}
\end{array}$$

Application of Homogeneous Dirichlet Conditions

- Returning to our WR statement and using $\underline{\bar{u}} = R^T\underline{u}$, we can now explicitly identify the global stiffness matrix A.

$$\mathcal{I} \;\; = \;\; \underline{\bar{v}}^T \bar{A} \underline{\bar{u}} \; = \; \underline{v}^T R \bar{A} R^T \underline{u} \; = \; \underline{v}^T A \underline{u}$$

- Recalling our earlier definitions, it's clear that we have

$$A \; = \; R\bar{A}R^T \; = \; RQ^T A_L Q R^T$$

- Note that, for all $u \in X_0^N \subset \mathcal{H}_0^1$, we have

$$\underline{u}_L \; = \; QR^T \underline{u}$$

- Q *ensures that* $u, v \in \mathcal{H}^1$.
- R^T *ensures that* $u, v \in \mathcal{H}_0^1$.

Summary: SEM Stiffness Matrix

Summarizing our matrix-vector product, we have:

$$\begin{aligned}
A\underline{u} &= RQ^T A_L Q R^T \underline{u} \\
&= RQ^T \begin{pmatrix} A^1 & & & \\ & A^2 & & \\ & & \ddots & \\ & & & A^E \end{pmatrix} QR^T \underline{u} \\
&= RQ^T \begin{pmatrix} A^1 & & & \\ & A^2 & & \\ & & \ddots & \\ & & & A^E \end{pmatrix} \begin{pmatrix} \underline{u}^1 \\ \underline{u}^2 \\ \vdots \\ \underline{u}^E \end{pmatrix}.
\end{aligned}$$

The *local* matrix-vector products (which determine the *physics*) are,

$$A^e \underline{u}^e|_i \quad = \quad \sum_{j=0}^{N} A^e_{ij}\, u^e_j \; = \; \sum_{j=0}^{N} \frac{2}{L^e} \left(\int_\Omega h_i'\, h_j'\, dr \right) u^e_j \, .$$

SEM Mass Matrix

Assuming (for now) that $f \in X_0^N$, construction of the r.h.s. follows in the same way. We have:

$$\begin{aligned}
B\underline{f} &= RQ^T B_L Q R^T \underline{f} \\
&= RQ^T \begin{pmatrix} B^1 & & & \\ & B^2 & & \\ & & \ddots & \\ & & & B^E \end{pmatrix} QR^T \underline{f} \\
&= RQ^T \begin{pmatrix} B^1 & & & \\ & B^2 & & \\ & & \ddots & \\ & & & B^E \end{pmatrix} \begin{pmatrix} \underline{f}^1 \\ \underline{f}^2 \\ \vdots \\ \underline{f}^E \end{pmatrix} .
\end{aligned}$$

Now the local matrix-vector products are

$$B^e \underline{f}^e \Big|_i = \sum_{j=0}^{N} B^e_{ij}\, f^e_j = \sum_{j=0}^{N} \frac{L^e}{2} \left(\int_{\hat{\Omega}} h_i\, h_j\, dr \right) f^e_j \,.$$

Correcting the RHS

Note: Our assumption of $\underline{f} \in X_0^N \subset \mathcal{H}_0^1$ is *way* too restrictive.

- In fact, it suffices to have $\underline{f} \in \mathcal{L}^2$, which allows jump discontinuites.

- Thus, we can lift the boundary condition (R) and continuity (Q) restrictions on $\underline{f}_L$ and simply write the r.h.s. as

$$\underline{g} \;=\; RQ^T B_L \underline{f}_L \;=\; RQ^T \begin{pmatrix} B^1 & & & \\ & B^2 & & \\ & & \ddots & \\ & & & B^E \end{pmatrix} \begin{pmatrix} \underline{f}^1 \\ \underline{f}^2 \\ \vdots \\ \underline{f}^E \end{pmatrix} .$$

- Notice that f is now happily in $\mathcal{L}^2$ as there is no Q nor R to apply.

- On the left, however, we still have Q and R because $v \in \mathcal{H}_0^1$.

- These terms are an important part of the projection process.

Final System of Equations

- So, we (finally!) arrive at our system of equations, where we now understand every detail:

$$\boxed{A\underline{u} = \underline{g}}$$

- Notice how, equipped with the *right tools*, the derivation of the r.h.s. was *much* (*much*) faster?!

Final System of Equations

- In all its detail, our solution (in local form) reads

$$\underline{u}_L = QR^T \left(RQ^T A_L Q R^T\right)^{-1} RQ^T B_L \underline{f}_L$$

- As we move to new physics, and to higher space dimensions, the final form will look much the same.

- The essential tools,

  — *local physics* $(A^e,\, B^e)$,

  — *appropriate continuity* (Q), and

  — *boundary conditions* (R),

and their interactions have been carefully established.

BREAK

SEM, Next Steps

- Lecture 2: 1D
  - GLL quadrature
  - Other BCs: Neumann
  - Advection
  - Nonlinear example

- Lecture 3: 2D and 3D
  - Matrix formulation
  - Curvilinear / mesh transformations
  - Preconditioned iterative solvers

Quadrature Rules for the SEM

- One of the primary reasons for choosing Gauss-Lobatto-Legendre points as nodal points is that they yield *well-conditioned* systems. (More on this point shortly.)

- It also allows us to significantly simplify operator evaluation, *especially in 3D*, which is where cost counts the most!

- Let’s begin with the stability (i.e., conditioning) issue.

Conditioning of the SEM Operators

- The condition number, κ, of a linear system governs the *round-off* error and, ultimately, the number of correct digits retained when multiplying a vector by a matrix or its inverse.

- For a symmetric positive definite (SPD) matrix A (as in our case), κ is the ratio of max to min eigenvalues:

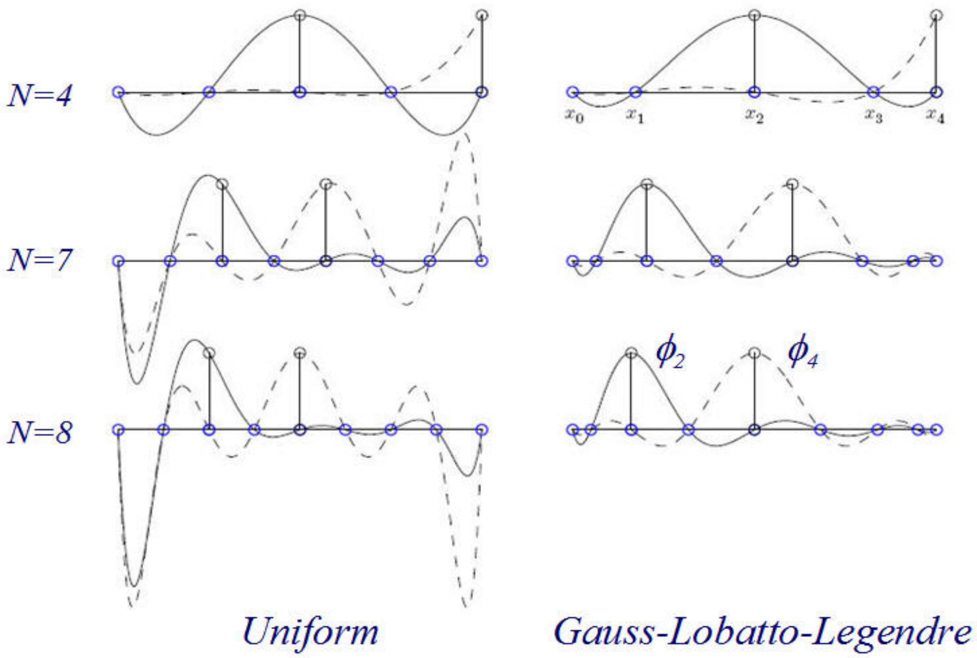
$$\kappa \sim \lambda_{\max} / \lambda_{\min}$$

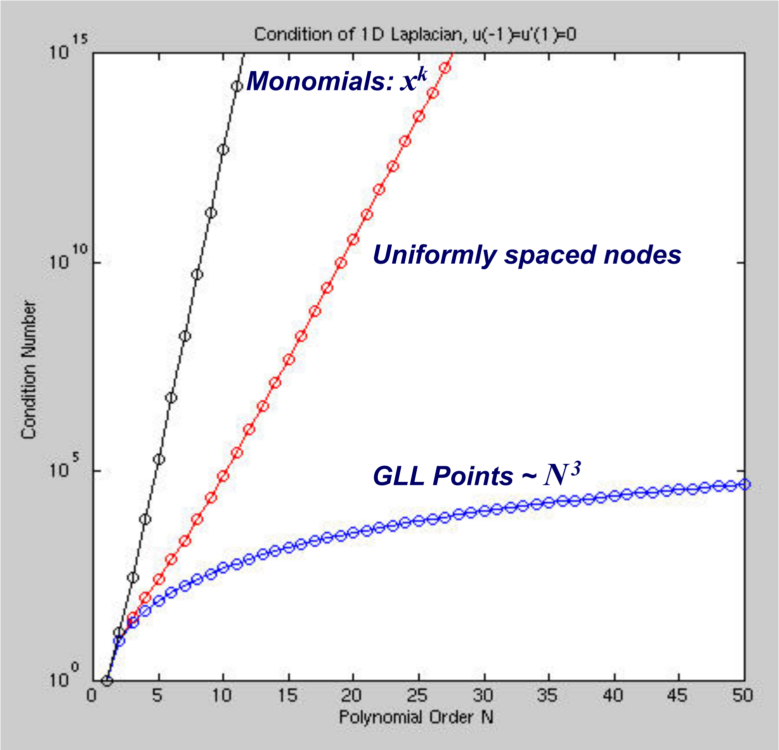
- If $\kappa \sim 10^k$, you can expect to lose ~ k digits when solving $A\underline{u} = \underline{g}$, so a smaller condition number is better.

- As indicated earlier, the condition number of A is governed by the choice of basis functions.

- In infinite-precision arithmetic, however, the choice is immaterial since the Galerkin scheme ensures that we would get the same *best-fit* solution.

Condition Number of A vs. Polynomial Order





- Monomials and Lagrange interpolants on uniform points exhibit *exponentional growth* in condition number.

- With just a 7x7 system the monomials would lose 10 significant digits (of 15, in 64-bit arithmetic).

Quadrature for the SEM

- In addition to good conditioning, use of nodal bases on the GLL points also results in reduced operator evaluation costs.

- Our local 1D stiffness and mass matrices on $\hat{\Omega}$ have the form

$$\hat{A}_{ij} \;=\; \int_{-1}^{1} \frac{dh_i}{dr}\,\frac{dh_j}{dr}\,dr, \qquad \hat{B}_{ij} \;=\; \int_{-1}^{1} h_i(r)\,h_j(r)\,dr.$$

- A key idea is to use Gauss quadrature to evaluate these integrals, either exactly or approximately.

- The Gauss-Legendre quadrature problem is stated as follows:

  Consider $q(x) \in \mathbb{P}_M$. Find points $\xi_k \in \hat{\Omega}$ and weights ρ_k, $k = 0, \dots, N$, such that

$$\sum_{k=0}^{N} \rho_k\, q(\xi_k) \;\equiv\; \int_{-1}^{1} q(x)\,dr \qquad \forall\, q \,\in\, \mathbb{P}_M,$$

  with M as large as possible.

What is the highest possible polynomial order, M?

- Let's look at the cardinality, $|\,.\,|$, of the sets.

$$\begin{aligned} |\,\mathbb{P}_M\,| &= M + 1 \\ |\,\rho_k\,| + |\,\xi_k\,| &= 2N + 2 \\ M+1 = 2N+2 \qquad &\Longleftrightarrow \qquad M = 2N+1 \end{aligned}$$

- Indeed, it is possible to find ξ_k and ρ_k such that *all polynomials* of degree $\leq M = 2N+1$ are integrated exactly.

- The ξ_k's are the zeros of the Legendre polynomial, $L_{N+1}(r)$.

- The ρ_k's are the integrals of the cardinal Lagrange polynomials passing through these points:.

$$\rho_k = \int_{-1}^{1} \tilde{h}_k(r)\,dr, \qquad \tilde{h}_k \in \mathbb{P}_N, \qquad \tilde{h}_k(\xi_j) = \delta_{kj}, \quad k = 0, \ldots, N$$

Gauss-Lobatto-Legendre Quadrature for the SEM

- This is the same idea as Gauss-Legendre (GL) quadrature, save that with GLL quadrature we *prescribe* two of the points: $\xi_0 := -1$ and $\xi_N := 1$.

- Now, we have only $2N + 2 - 2 = 2N$ degrees of freedom.

- So, we expect $|\mathbb{P}_M| = M + 1 = 2N - 1 \longrightarrow \boxed{M = 2N - 1.}$

- In this case, the ξ_k's are the zeros of $(1 - r^2)L'_N(r)$.

- As before, the ρ_k's are the integrals of the cardinal Lagrange polynomials passing through these points:.

$$\rho_k = \int_{-1}^{1} h_k(r)\, dr, \qquad h_k \in \mathbb{P}_N, \qquad h_k(\xi_j) = \delta_{kj}.$$

- Let's return to the local integrals in our WRT!

Quadrature for the SEM

- We now replace the integrals with quadrature.
- For the mass matrix, we have

$$\hat{B}_{ij} \quad = \quad \int_{-1}^{1} h_i(r)\, h_j(r)\, dr \;\approx\; \sum_{k=0}^{N} \rho_k\, h_i(\xi_k)\, h_j(\xi_k) \;\equiv\; \rho_i\, \delta_{ij}$$

The last equivalence establishes the important result that B is *diagonal.*

Quadrature for the SEM

- In fact, it is always possible to construct such a diagonal mass matrix.
  - Simply start with a standard mass matrix and replace it by a diagonal matrix having the same *row sum* as the original.
  - This is often called *mass lumping*.
  - The rule of thumb with quadrature is to ensure that the error is small and that the resultant discrete operator has the correct spectral properties (e.g., care is required for *convection* operators).

- What is key for the SEM is that it is a *very good* diagonal mass matrix because of the *high order*, N.
  - The quadrature errors decay exponentially for smooth integrands.

Quadrature for the SEM

- For the stiffness matrix, we have

$$\hat{A}_{ij} \quad = \quad \int_{-1}^{1} \frac{dh_i}{dr}\,\frac{dh_j}{dr}\,dr \equiv \sum_{k=0}^{N} \rho_k \left.\frac{dh_i}{dr}\right|_{\xi_k} \left.\frac{dh_j}{dr}\right|_{\xi_k} \quad = \hat{D}^T \hat{B} \hat{D},$$

where $\hat{D}$ is the *derivative matrix* with entries

$$\hat{D}_{ij} \quad := \quad \left.\frac{dh_j}{dr}\right|_{\xi_i} .$$

- This result is *exact* because the integrand is a polynomial of degree $2N-2$.

- As we'll see shortly, quadrature is very convenient (but not exact) for *variable coefficient* problems.

Quadrature for the SEM

If we have a *variable coefficient* problem, e.g.,

$$-\frac{d}{dx}\, p(x)\, \frac{du}{dx} \;=\; f(x), \qquad u(0) = u(1) = 0,$$

then, after integration by parts, the local stiffness matrix is

$$A^e_{ij} \;=\; \frac{2}{L^e} \int_{-1}^{1} p^e(r)\, \frac{dh_i}{dr}\, \frac{dh_j}{dr}\, dr \;\approx\; \sum_{k=0}^{N} p^e_k\, \rho_k\, \left.\frac{dh_i}{dr}\right|_{\xi_k} \left.\frac{dh_j}{dr}\right|_{\xi_k} .$$

where $p^e_k := p(x^e_k)$.

- Let $P^e := diag(p^e_k)$. Then $A^e \;=\; \dfrac{2}{L^e}\, \hat{D}^T\, P^e\, \hat{B}\, \hat{D}\, .$
- Compare this with the standard case: $A^e \;=\; \dfrac{2}{L^e}\, \hat{D}^T\, \hat{B}\, \hat{D}\, .$
- Recall, $\hat{B}$ is diagonal.
- The approach outlined here is similar to *collocation*.

Let's Look at Some Examples

Let's take the variable coefficient problem

$$-\frac{d}{dx}\,p(x)\,\frac{du}{dx} \;=\; f(x), \qquad u(0) = u(\pi) = 0,$$

with $p(x) = e^x$ and exact solution $\tilde{u}(x) = \sin(x) \in \mathcal{H}^1_0$.

- The rhs in this case is $f(x) = e^x(\sin(x) - \cos(x))$.

- For this 1D example, we will form $A_L := \textit{block-diag}(A^e)$ on an element-by-element basis.

- We will then assemble it and restrict it to yield $A = RQ^T A_L Q R^T$.

- We then set up the rhs, $\underline{g} = RQ^T B_L f_L$ and solve $\underline{u} = A \backslash \underline{g}$ (matlab notation).

- We plot $(\underline{x}_L, \underline{u}_L)$ and $(\underline{x}_L, \underline{\tilde{u}}_L)$ using *local* coordinates.

- Finally, we check the pointwise error.

var_1d_poission.m

Convection-Diffusion Example

Consider the steady-state convection-diffusion equation,

$$-\nu \frac{\partial^2 \tilde{u}}{\partial x^2} + c\,\frac{\partial \tilde{u}}{\partial x} \;=\; f, \qquad u(0) = u(1) = 0.$$

- The WR formulation reads: Find $\tilde{u} \in \mathcal{H}_0^1$ such that

$$\int_\Omega \frac{dv}{dx}\,\nu\,\frac{d\tilde{u}}{dx}\,dx \;+\; \int_\Omega v\,c\,\frac{d\tilde{u}}{dx}\,dx \;=\; \int_\Omega v\,f\,dx \qquad \forall v \;\in\; \mathcal{H}_0^1.$$

- Discretization proceeds by seeking $u \in X_0^N \subset \mathcal{H}_0^1$ such that

$$\int_\Omega \frac{dv}{dx}\,\nu\,\frac{du}{dx}\,dx \;+\; \int_\Omega v\,c\,\frac{du}{dx}\,dx \;=\; \int_\Omega v\,f\,dx \qquad \forall v \;\in\; X_0^N.$$

Convection-Diffusion Example

There is only one new component:

$$c(v,u) \;:=\; \int_{\Omega} v\, c\, \frac{du}{dx}\, dx\,.$$

- This leads to the local *convection matrix*,

$$C^e_{ij} \;:=\; \int_{\Omega^e} c(x)\, \phi_i(x)\, \frac{d\phi_j}{dx}\, dx\,.$$

- Switching to the reference element, $\hat{\Omega} := [-1,1]$,

$$\begin{aligned} C^e_{ij} \;&=\; \int_{\hat{\Omega}} c^e(r)\, h_i(r)\, \frac{dh_j}{dr}\, dr\,, \qquad c^e(r) \;:=\; c(x^e(r)) \\ &\approx\; \sum_{k=0}^{N} \rho_k\, c^e(\xi_k)\, h_i(\xi_k)\, \left.\frac{dh_j}{dr}\right|_{\xi_k} \\ &=\; c^e_i\, \rho_i\, \hat{D}_{ij}\,. \end{aligned}$$

- If $c^e := diag(c^e_j)$ is the diagonal matrix of local velocity values, then

$$\boxed{C^e = c^e \hat{B} \hat{D},}$$

Convection-Diffusion System of Equations

- The full system for the convection-diffusion equation reads

$$\begin{array}{rcl} \boxed{(A + C)\,\underline{u} \;=\; RQ^T B_L \underline{f},} & & \text{with} \\ A & = & RQ^T A_L Q R^T \,, \\ C & = & RQ^T C_L Q R^T \,. \end{array}$$

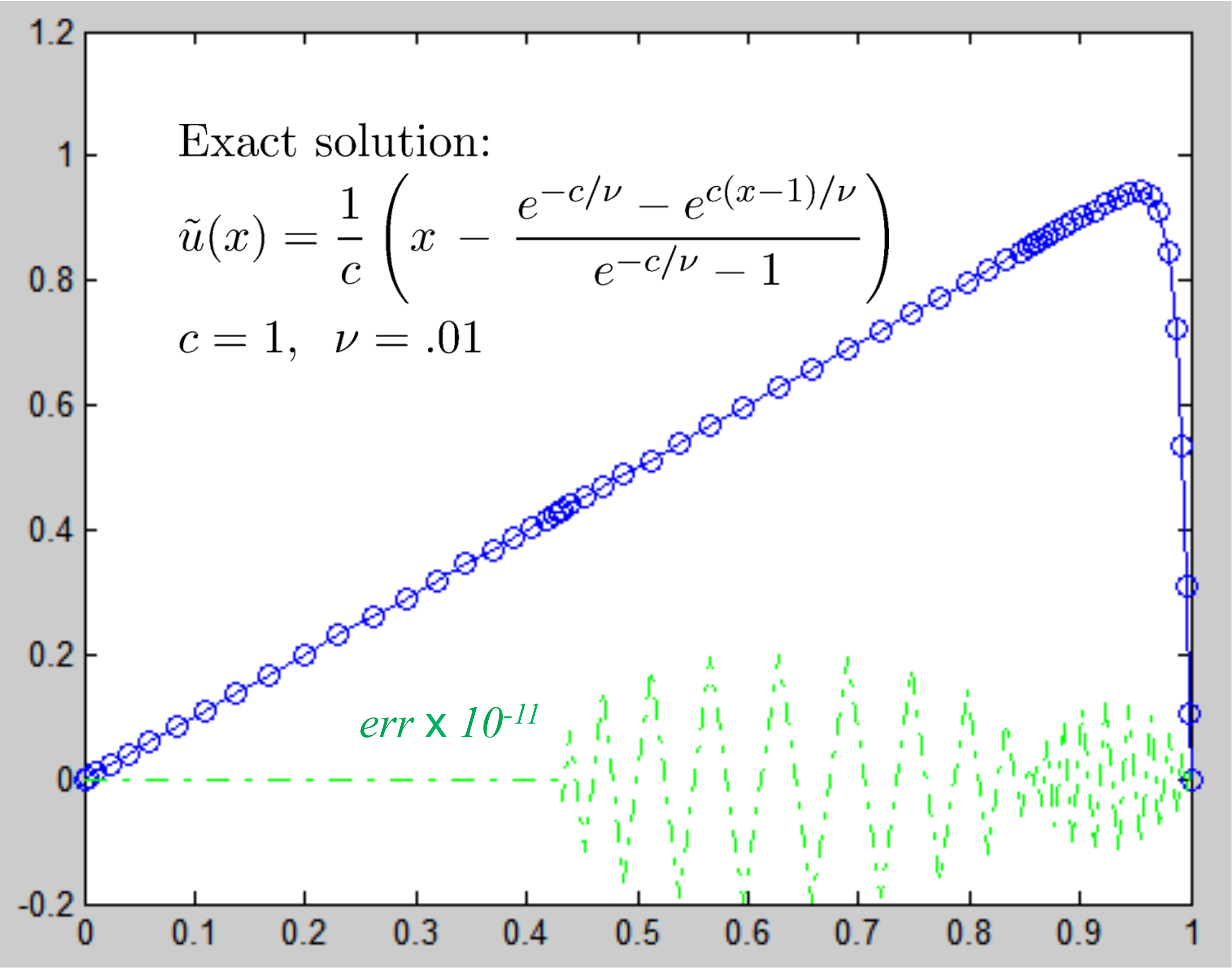
Here, A_L:=*block-diag*(A^e) and C_L:=*block-diag*(C^e).

- If ν is constant, then $A^e = \frac{2\nu}{L^e}\hat{A}$. For variable ν, we have

$$A^e \;=\; \frac{2}{L^e}\hat{D}^T\,\nu^e\,\hat{B}\hat{D}, \qquad \nu^e := diag(\nu^e_j)\,.$$

- **Q:** Why is it that A^e depends on L^e, but C^e does not?

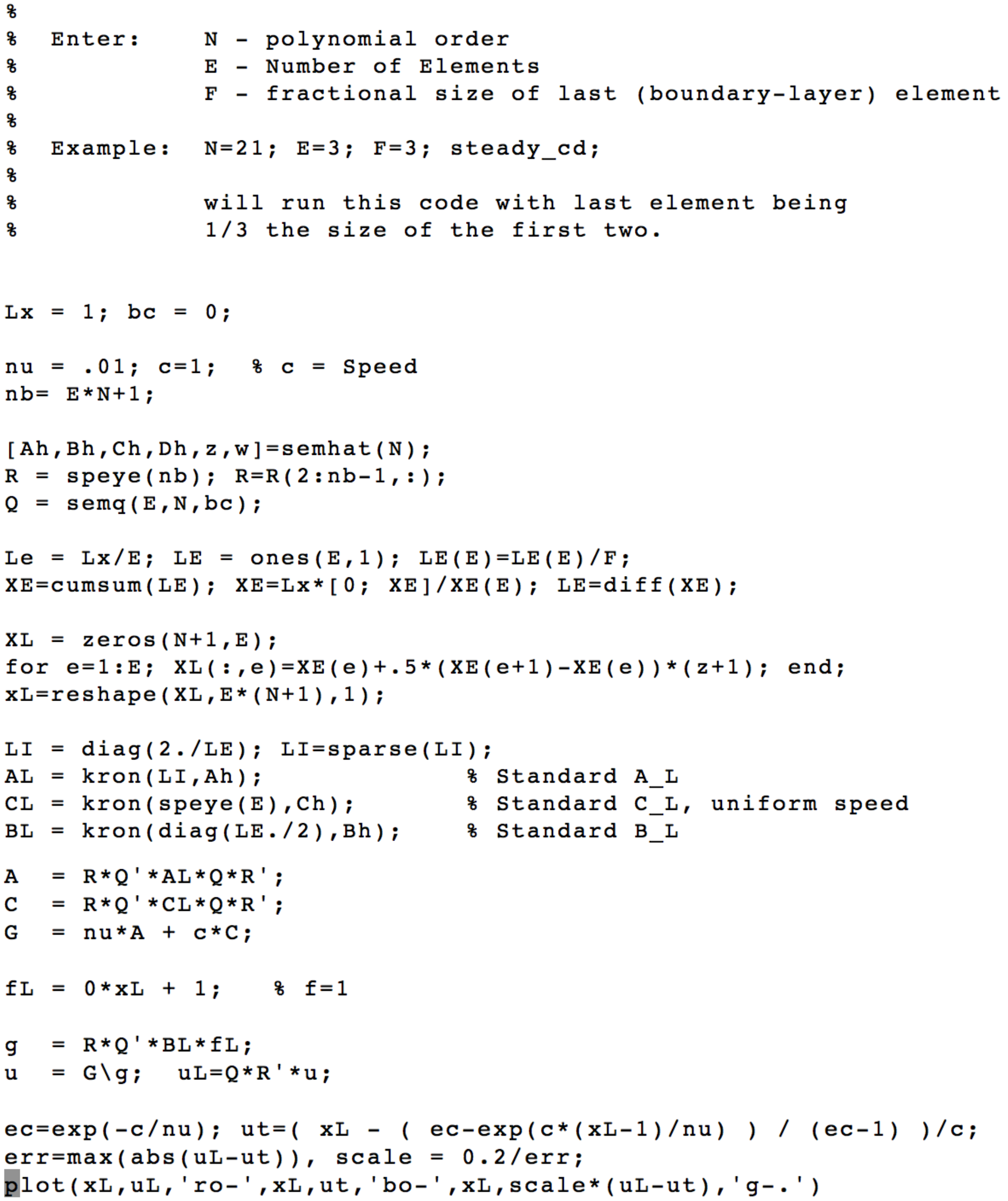
CD: Solution and Error, $\nu = 10^{-2}$, E=3, N=21



- Here, in order to resolve the boundary layer, the last element is 1/3 the size of the others. The error is 2.e-12.

steady_1d_cd.m

Matlab Code for Steady State Convection-Diffusion



```
%
%  Enter:     N - polynomial order
%             E - Number of Elements
%             F - fractional size of last (boundary-layer) element
%
%  Example:   N=21; E=3; F=3; steady_cd;
%
%             will run this code with last element being
%             1/3 the size of the first two.


Lx = 1; bc = 0;

nu = .01; c=1;  % c = Speed
nb= E*N+1;

[Ah,Bh,Ch,Dh,z,w]=semhat(N);
R = speye(nb); R=R(2:nb-1,:);
Q = semq(E,N,bc);

Le = Lx/E; LE = ones(E,1); LE(E)=LE(E)/F;
XE=cumsum(LE); XE=Lx*[0; XE]/XE(E); LE=diff(XE);

XL = zeros(N+1,E);
for e=1:E; XL(:,e)=XE(e)+.5*(XE(e+1)-XE(e))*(z+1); end;
xL=reshape(XL,E*(N+1),1);

LI = diag(2./LE); LI=sparse(LI);
AL = kron(LI,Ah);              % Standard A_L
CL = kron(speye(E),Ch);        % Standard C_L, uniform speed
BL = kron(diag(LE./2),Bh);     % Standard B_L

A   = R*Q'*AL*Q*R';
C   = R*Q'*CL*Q*R';
G   = nu*A + c*C;

fL  = 0*xL + 1;    % f=1

g   = R*Q'*BL*fL;
u   = G\g;   uL=Q*R'*u;

ec=exp(-c/nu); ut=( xL - ( ec-exp(c*(xL-1)/nu) ) / (ec-1) )/c;
err=max(abs(uL-ut)), scale = 0.2/err;
plot(xL,uL,'ro-',xL,ut,'bo-',xL,scale*(uL-ut),'g-.')
```

⋮

Matlab Demo: steady_1d_cd.m

- What happens when we vary ν ?

- Try small ν for n even and n odd. Is there any significant difference?

- For small ν, can you refine your mesh (h, p, or r refinement) to recover a good solution?

- Exercise for later:
  - Examine the behavior when you *time-march* the solution to a steady state, both with and without a stabilizing filter.
  - What is the impact of the filter in the well-resolved case?

Inhomogeneous Neumann Condition (1/4)

Example:
$$-\frac{d^2\tilde{u}}{dx^2} = f(x) \qquad \begin{cases} \tilde{u}(-1) = 0 \\ \tilde{u}'(1) = g \end{cases}$$

Standard Derivation: *Find* $u \in X_0^N$ *such that*

$$-\int_{-1}^{1} v\,\frac{d^2u}{dx^2}\,dx = \int_{-1}^{1} v\,f(x)\,dx \;\; \forall\, v \in X_0^N$$

Integrate by parts and use inhomogeneous Neumann condition:

$$\int_{-1}^{1} \frac{dv}{dx}\,\frac{du}{dx}\,dx - v\,\frac{du}{dx}\bigg|_{-1}^{1} = \int_{-1}^{1} v\,f(x)\,dx$$

$$\int_{-1}^{1} \frac{dv}{dx}\,\frac{du}{dx}\,dx - \left[v(1)g - \cancel{v(-1)\frac{du}{dx}\bigg|_{-1}} \right] = \int_{-1}^{1} v\,f(x)\,dx$$

$$\int_{-1}^{1} \frac{dv}{dx}\,\frac{du}{dx}\,dx = \int_{-1}^{1} v\,f(x)\,dx + v(1)g.$$

Inhomogeneous Neumann Condition (2/4)

- This equation is standard, save for the addition of the extra term at $x{=}1$:

$$\int_{-1}^{1} \frac{dv}{dx} \frac{du}{dx} \, dx \;\; = \;\; \int_{-1}^{1} v \, f(x) \, dx \; + \; v(1) g \;\; \forall v \in X_0^N .$$

- Consider now N equations, generated by taking for $i = 1, \ldots, N$,

$$v(x) = \phi_i(x) = h_i(x),$$

where we are taking the basis functions to be the cardinal Lagrange polynomials, $h_i(x)$ for $x \in [-1, 1] = \hat{\Omega}$.

- Note that all basis functions vanish at $x = 1$ except for $h_N(x)$, so only the **last equation** is modified.

Inhomogeneous Neumann Condition (3/4)

- Only the **last equation** is modified.

$$\int_{-1}^{1} \frac{dv}{dx}\frac{du}{dx}\,dx \;=\; \int_{-1}^{1} v\,f(x)\,dx \;+\; v(1)g \;\;\forall v \in X_0^N.$$

- Accounting for the homogeneous Dirichlet condition at $x = -1$,

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & & & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}; \;=\; \begin{pmatrix} \rho_1 f_1 \\ \rho_2 f_2 \\ \vdots \\ \rho_N f_N \;+\; g \end{pmatrix}.$$

- Though not immediately evident, we this system implies:

$$\lim_{N\longrightarrow\infty} \left.\frac{du}{dx}\right|_{x=1} \;=\; g$$

- To see this, we start with the top equation and integrate back.

Inhomogeneous Neumann Condition (4/4)

- Integrating the LHS of

$$\int_{-1}^{1} \frac{dv}{dx}\,\frac{du}{dx}\,dx \;=\; \int_{-1}^{1} v\,f(x)\,dx \;+\; v(1)g \;\;\forall v \in X_0^N .$$

  by parts and rearranging, we obtain

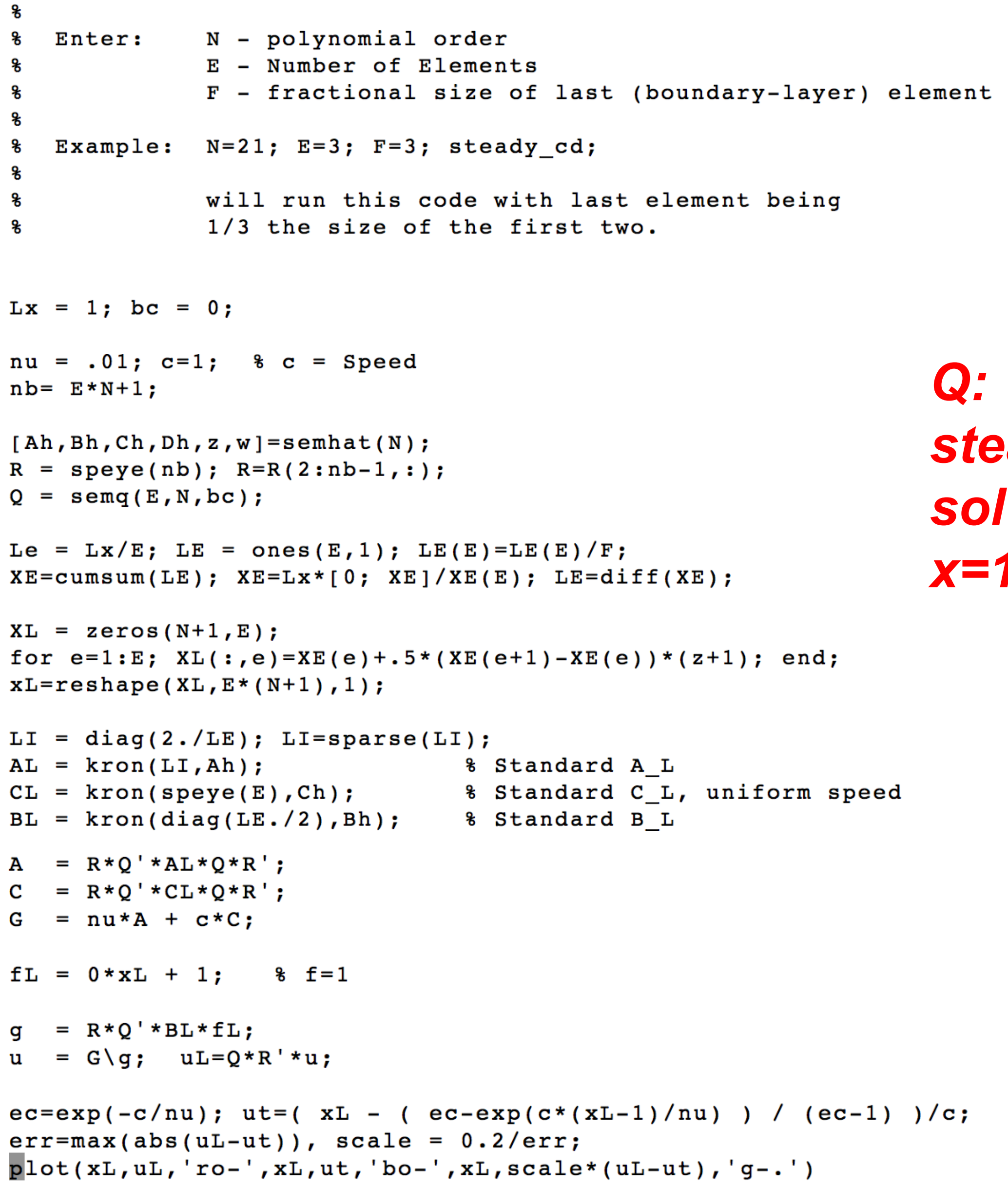
$$\int_{-1}^{1} v\left[-\frac{d^2u}{dx^2} - f\right] dx \;+\; v\,\frac{du}{dx}\bigg|_{x=1} \;=\; v(1)g \;\;\forall v \in X_0^N .$$

- Because our quadrature rule is exact for the 2nd-order term, the last equation of the preceding slide reads

$$\rho_N\left[-\frac{d^2u}{dx^2} - f\right] \;+\; v\,\frac{du}{dx}\bigg|_{x=1} \;=\; v(1)g.$$

- Since $\rho_N \sim 2N^{-2}$ we have: $\displaystyle\lim_{N\longrightarrow\infty} \frac{du}{dx}\bigg|_{x=1} \;=\; g.$

Modify Steady State Convection-Diffusion



```
%
%  Enter:     N - polynomial order
%             E - Number of Elements
%             F - fractional size of last (boundary-layer) element
%
%  Example:   N=21; E=3; F=3; steady_cd;
%
%             will run this code with last element being
%             1/3 the size of the first two.


Lx = 1; bc = 0;

nu = .01; c=1;  % c = Speed
nb= E*N+1;

[Ah,Bh,Ch,Dh,z,w]=semhat(N);
R = speye(nb); R=R(2:nb-1,:);
Q = semq(E,N,bc);

Le = Lx/E; LE = ones(E,1); LE(E)=LE(E)/F;
XE=cumsum(LE); XE=Lx*[0; XE]/XE(E); LE=diff(XE);

XL = zeros(N+1,E);
for e=1:E; XL(:,e)=XE(e)+.5*(XE(e+1)-XE(e))*(z+1); end;
xL=reshape(XL,E*(N+1),1);

LI = diag(2./LE); LI=sparse(LI);
AL = kron(LI,Ah);              % Standard A_L
CL = kron(speye(E),Ch);        % Standard C_L, uniform speed
BL = kron(diag(LE./2),Bh);     % Standard B_L

A  = R*Q'*AL*Q*R';
C  = R*Q'*CL*Q*R';
G  = nu*A + c*C;

fL = 0*xL + 1;    % f=1

g  = R*Q'*BL*fL;
u  = G\g;   uL=Q*R'*u;

ec=exp(-c/nu); ut=( xL - ( ec-exp(c*(xL-1)/nu) ) / (ec-1) )/c;
err=max(abs(uL-ut)), scale = 0.2/err;
plot(xL,uL,'ro-',xL,ut,'bo-',xL,scale*(uL-ut),'g-.')
```

⋮

Q: How should we modify the steady state convection-diffusion solver for a Neumann condition at x=1 ?

steady_1d_cd.m ?

Unsteady Convection-Diffusion Example

Unsteady Convection-Diffusion Example

We now have the time-dependent problem,

$$\frac{\partial \tilde{u}}{\partial t} + c\,\frac{\partial \tilde{u}}{\partial x} = \nu \frac{\partial^2 \tilde{u}}{\partial x^2} + f,$$

$$u(0,t) = u(1,t) = 0, \qquad u(x,0) = u_0(x).$$

- Rearrange and evaluate each term at time $t^m = m\Delta t$,

$$\left.\frac{\partial \tilde{u}}{\partial t}\right|_{t^m} - \nu \left.\frac{\partial^2 \tilde{u}}{\partial x^2}\right|_{t^m} = \left(f - c\,\frac{\partial \tilde{u}}{\partial x} \right)_{t_m} .$$

- The first term we evaluate using kth-order *backward difference formulae* (BDFk),

$$\begin{aligned}
\left.\frac{\partial \tilde{u}}{\partial t}\right|_{t^m} &= \frac{u^m - u^{m-1}}{\Delta t} + O(\Delta t) \\
&= \frac{3u^m - 4u^{m-1} + 3u^{m-2}}{2\Delta t} + O(\Delta t^2) \\
&= \frac{11u^m - 18u^{m-1} + 9u^{m-2} - 2u^{m-3}}{6\Delta t} + O(\Delta t^3)
\end{aligned}$$

BDFk Formulas: GTE = $O(\Delta t^k)$

$$\text{BDF1:} \quad \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{u^n - u^{n-1}}{\Delta t} + O(\Delta t)$$

$$\text{BDF2:} \quad \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{3u^n - 4u^{n-1} + u^{n-2}}{2\Delta t} + O(\Delta t^2)$$

$$\text{BDF3:} \quad \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{11u^n - 18u^{n-1} + 9u^{n-2} - 2u^{n-3}}{6\Delta t} + O(\Delta t^3).$$

- k-th order accurate
- Implicit
- Unconditionally stable only for k $\leq$ 2
- Multi-step: require data from previous timesteps

Unsteady Convection-Diffusion Example

- The viscous term is treated *implicitly*,

$$\nu \frac{\partial^2 \tilde{u}}{\partial x^2}\bigg|_{t^m} = \nu \frac{d^2 \tilde{u}^m}{dx^2}$$

- Convection and the forcing are treated *explicitly* via extrapolation,

$$\begin{aligned}
\left(f - c\,\frac{\partial \tilde{u}}{\partial x}\right)_{t_m} &= \left(f^{m-1} - c\,\frac{d\tilde{u}^{m-1}}{dx}\right) + O(\Delta t) \\
&= 2\left(f^{m-1} - c\,\frac{d\tilde{u}^{m-1}}{dx}\right) - \left(f^{m-2} - c\,\frac{d\tilde{u}^{m-2}}{dx}\right) + O(\Delta t^2) \\
&= 3\left(f^{m-1} - c\,\frac{d\tilde{u}^{m-1}}{dx}\right) - 3\left(f^{m-2} - c\,\frac{d\tilde{u}^{m-2}}{dx}\right) \\
&+ \left(f^{m-3} - c\,\frac{d\tilde{u}^{m-3}}{dx}\right) + O(\Delta t^3)
\end{aligned}$$

Spatial – Temporal Discretization

For simplicity, we consider BDF2/EXT2 and discretize in space with the WRT. To begin, we arrange terms.

We use,

$$\left.\frac{\partial \tilde{u}}{\partial t}\right|_{t^m} \approx \frac{3u^m - 4u^{m-1} + 3u^{m-2}}{2\Delta t}$$

$$g^m \approx 2g^{m-1} - g^{m-2}, \qquad g^k := \left(f - c\frac{\partial u}{\partial x}\right)^k,$$

- When rearranged, we have our $O(\Delta t^2)$ timestepping scheme:

$$\frac{3}{2}u^m - \nu\Delta t\frac{d^2u^m}{dx^2} = \frac{4}{2}u^{m-1} - \frac{1}{2}u^{m-2} + 2g^{m-1} - g^{m-2} := q^m$$

- Proceding as before with the WRT, the fully-discretized problem reads,

Find $u^m \in X_0^N$ such that

$$\frac{3}{2}(v, u^m)_N + \nu\Delta t\, a_N(v, u^m) = (v, q^m)_N,$$

where we have used subscript N to denote the use of discrete GLL quadrature.

Spatial – Temporal Discretization

- Inserting our basis functions and coefficient vectors as before we have,

$$\frac{3}{2}\underline{v}^T B \underline{u}^m + \nu \Delta t \, \underline{v}^T A \underline{u}^m \;=\; \underline{v}^T Q^T B_L \underline{q}_L^m, \qquad \forall \; \underline{v} \in \mathbb{R}^n.$$

Or simply,

$$H \underline{u}^m \;=\; Q^T B_L \underline{q}_L^m,$$

where the discrete Helmholtz operator is defined as

$$H = B + \nu \Delta t \, A.$$

- Recall that B is diagonal for the SEM and that, typically, $\nu \Delta t \ll 1$, so that H is strongly diagonally dominant, which tends to make H (or, more properly, $B^{-1}H$) well-conditioned.

- This property is very helpful when solving time-dependent problems using iterative solvers in higher-space dimensions.

Additional Timestepping Considerations

- We typically use the 3rd-order schemes as their stability diagram encompasses part of the imaginary axis, which is where the eigenvalues of convection-dominated systems are found.

- The BDF3 and EXT3 formulae require prior values of u and the data, so we typically start with BDF1, 2, ... , 3, which means we are at best $O(\Delta t^2)$ accurate. (Why?)

- For turbulence, this generally doesn't matter because the initial conditions are contrived. For restarted solutions it's a bit annoying – one can always save multiple solutions for restart, which is our approach, or switch to an RK scheme for start-up.

Additional Timestepping Considerations

- A very important question for explicit or semi-implicit timesteppers is, *How large a timestep Δt can one take and still be stable?*

- This question depends jointly on the *temporal* and *spatial* discretizations.

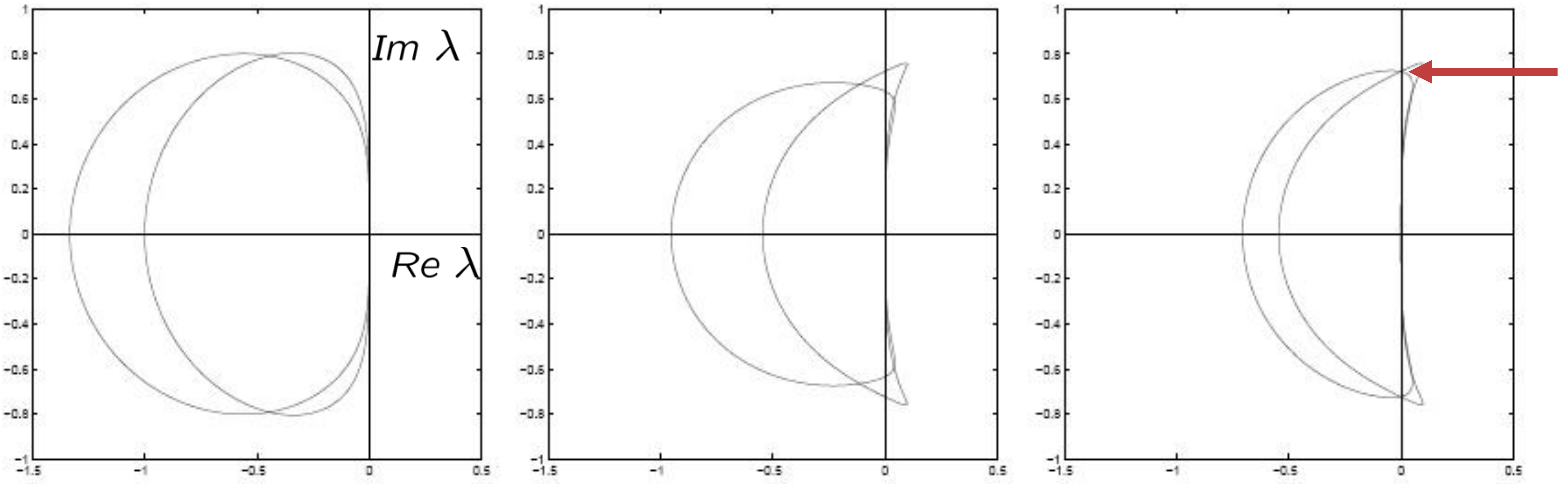
- Specifically, the temporal discretization determines the region of stability for the explicit (or implicit) timestepper.

- The spatial discretizaton determines the eigenvalues for which $\lambda \Delta t$ must fit inside the stability region.

- Starting with the stability region, we show how to rapidly estimate these quantities to ensure a stable timestepper.

Stability of Various Timesteppers

- Derived from model problem $\frac{du}{dt} = \lambda u$
- Stability regions shown in the λΔt plane (stable *inside* the curves)



~ 0.72

Figure 1: Stability regions for (left) AB2 and BDF2/EXT2, (center) AB3 and BDF3/EXT3, and (right) AB3 and BDF2/EXT2a.

- To make effective use of this plot, we need to know something about the eigenvalues λ of the discrete convection operator.
- But first, *How are these plots generated?*

Determining the Neutral-Stability Curve

Consider BDF2/EXT2, and apply it to $\dfrac{du}{dt} \;=\; \lambda\, u$:

$$3u^m \;-\; 4u^{m-1} \;+\; u^{m-2} \;\;=\;\; 2\lambda\Delta t\,\left(2u^{m-1} - u^{m-2}\right).$$

Seek solutions of the form $u^m = (z)^m$, $z \in C$:

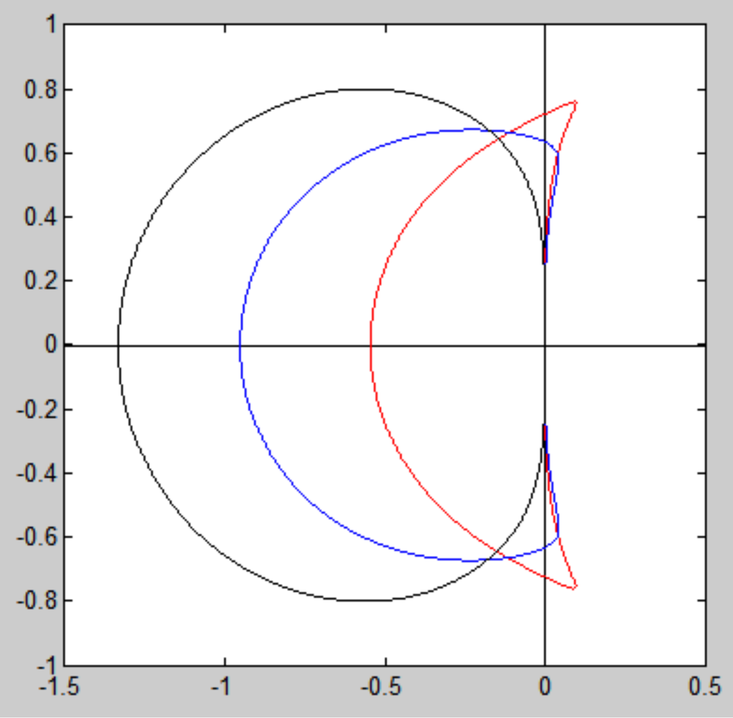
$$\begin{aligned} 3z^m \;-\; 4z^{m-1} \;+\; z^{m-2} \;\;&=\;\; 2\lambda\Delta t\,\left(2z^{m-1} - z^{m-2}\right). \\ 3z^2 \;-\; 4z \;+\; 1 \;\;&=\;\; 2\lambda\Delta t\,(2z - 1)\,. \end{aligned}$$

Set $z = e^{i\theta}$, $\theta \in [0, 2\pi]$, and solve for $\lambda\Delta t$:

$$\lambda\Delta t \;\;=\;\; \frac{3e^{i2\theta} \;-\; 4e^{i\theta} \;+\; 1}{2\,(2e^{i\theta} - 1)}.$$

Matlab Code: stab.m

```
ymax=1; ep=1.e-13; yaxis=[-ymax*ii ymax*ii]’;  % Plot axes
xaxis=[-2.0+ep*ii 2.0+ep*ii]’;
hold off; plot (yaxis,’k-’); hold on; plot (xaxis,’k-’);
axis square; axis([-ymax-.5 ymax-.5 -ymax ymax]);

ii=sqrt(-1); th=0:.001:2*pi;  th=th’; ith=ii*th; ei=exp(ith);
E = [ ei 1+0*ei 1./ei 1./(ei.*ei) 1./(ei.*ei.*ei)];


ab0   = [1 0.0 0.0 0. 0.]’;
ab1   = [0 1.0 0.0 0. 0.]’;
ab2   = [0 1.5 -.5 0. 0.]’;
ab3   = [0 23./12. -16./12. 5./12. 0.]’;
bdf1  = (([ 1.  -1.   0.   0. 0.])/1.)’;
bdf2  = (([ 3.  -4.   1.   0. 0.])/2.)’;
bdf3  = (([11. -18.   9. -2. 0.])/6.)’;
exm   = [1 0   0 0 0]’;
ex1   = [0 1   0 0 0]’;
ex2   = [0 2 -1 0 0]’;
ex3   = [0 3 -3 1 0]’;
du    = [1. -1. 0. 0. 0.]’;
```



```
ldtab3 =(E*du)./(E*ab3);   plot (ldtab3 ,’r-’);  % AB3
bdf3ex3=(E*bdf3)./(E*ex3); plot (bdf3ex3,’b-’);  % BDF3/EXT3
bdf2ex2=(E*bdf2)./(E*ex2); plot (bdf2ex2,’k-’);  % BDF2/EXT2
```

Relating Stability Region to Δt

- From the stability curves, we know the limits on $\lambda \Delta t$.

- For 2nd-order centered finite-differences, we know that the maximum (modulus) eigenvalue of the first-order operator $(u_{j+1} - u_{j-1})/\Delta x)$ is $\pm i$.

- This gives rise to the well-known CFL condition

$$\max_k |\lambda_k \Delta t| = \frac{c\Delta t}{\Delta x} \leq 0.72... \qquad \text{(for AB3).}$$

- In effect, the CFL number is a measure of the maximum modulus eigenvalue of the convective operator. Here, we define it as:

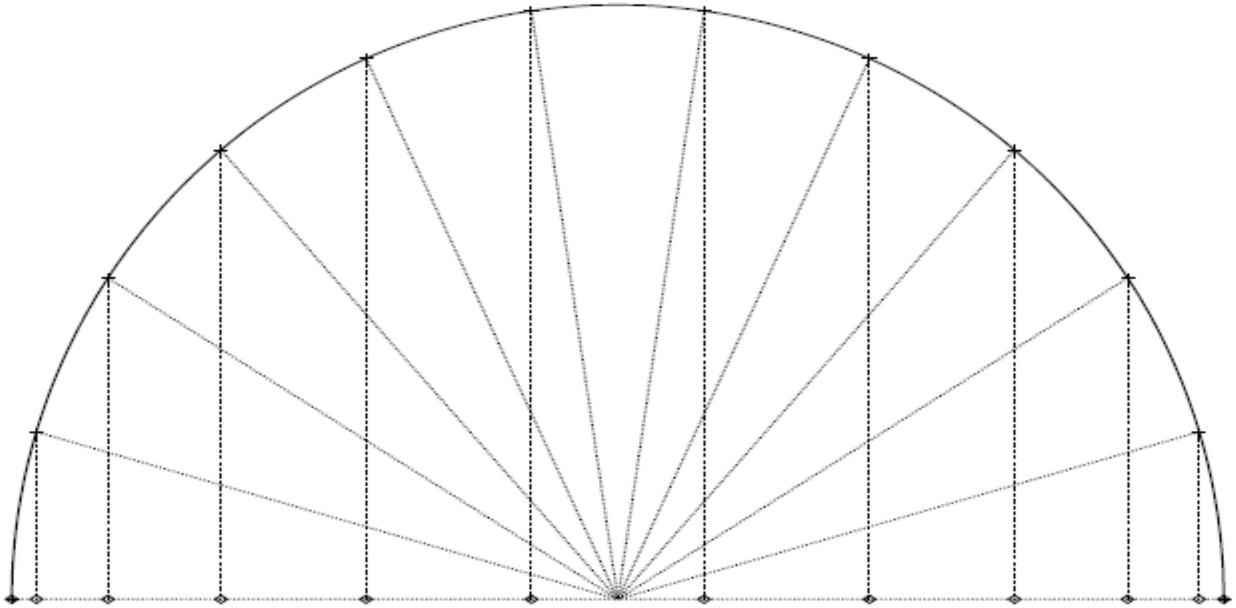
$$CFL := \max_j \frac{c_j \Delta t}{\Delta x_j},$$

where c_j is the local velocity and Δx_j is the local grid spacing.

- Note that CFL is readily computable and gives a very accurate estimate of $\max |\lambda \Delta t|$.

Relating Stability Region to Δt

- For the SEM, we use the same definition of CFL. In this case, however, we have to consider the minimum space of the GLL points, which have a spacing similar to the Gauss-Lobatto Chebyshev (GLC) points shown below.



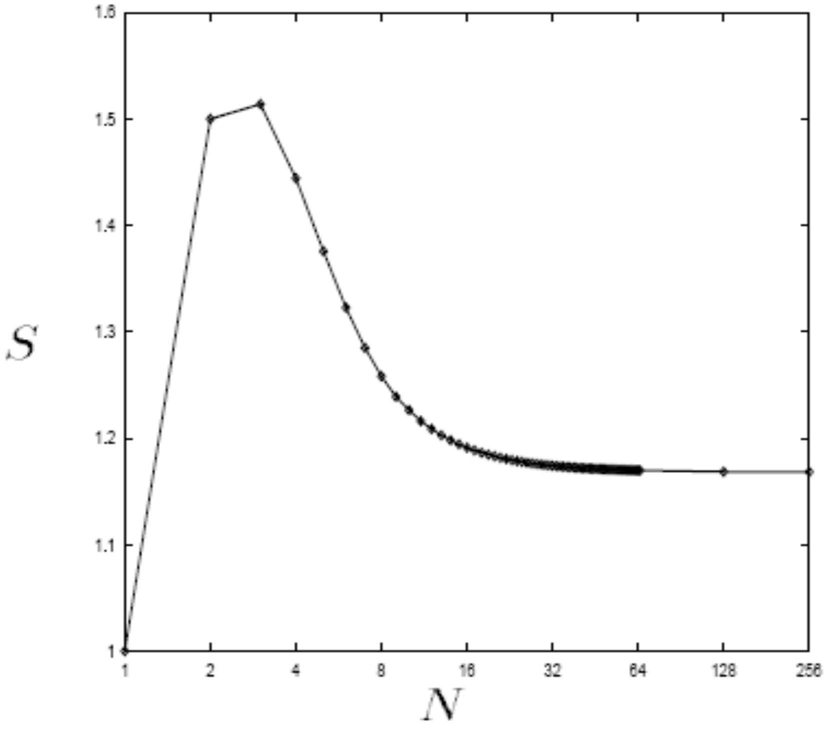
$$\Delta x_{\max} \sim \frac{\pi}{N} \qquad \Delta x_{\min} \sim \frac{\pi^2}{2N^2}$$

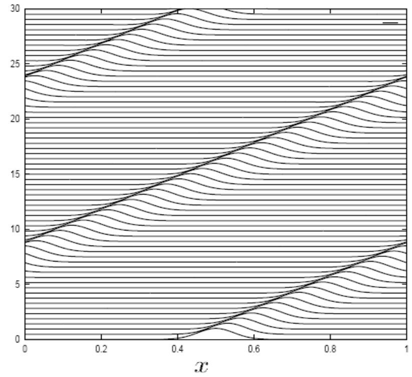
Relating Stability Region to Δt

- It turns out that $\max|\lambda|$ for the SEM is a bit larger than $c/\Delta t_{\text{min}}$ by a factor $1.16 \leq\ S\ \leq 1.5$, which is plotted below as a function of polynomial order.

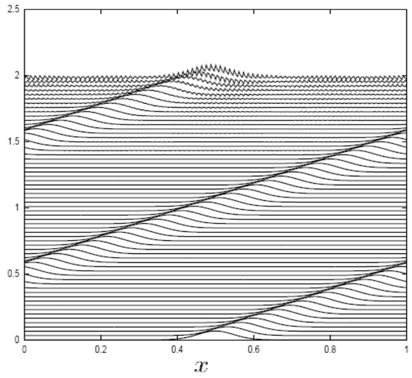
- Thus, say for AB3, we need $\quad \max|\lambda|\Delta t \ \sim\ S\dfrac{c\Delta t}{\Delta x} \ \leq 0.72.$

- The consequences of not meeting this condition are seen in the bottom-right plot of a traveling wave solution that is starting to blow up.





$$CFL\ <\ \frac{0.72}{S}$$



$$CFL\ >\ \frac{0.72}{S}$$

Unsteady Convection-Diffusion Example

From the preceding analysis, we can develop an unsteady convection-diffusion solver.

- **Semi-implicit update step:**

$$\underline{\hat{u}}_L = -\sum_{j=1}^{k} \beta_j \, \underline{u}_L^{n-j}$$

BDF terms

$$\underline{\hat{f}}_L = -\Delta t \sum_{j=1}^{k} \alpha_j \, cC\underline{u}_L^{n-j}$$

Extrapolated convection term

$$H\underline{u}^n = R\,Q^T \left(B_L \underline{\hat{u}}_L + \underline{\hat{f}}_L \right)$$

Implicit solve

$$\underline{\hat{u}}_L^n = Q\,R^T \,\underline{u}^n,$$

Map back to local form

  where

$$H := \beta_0 \, B + \nu \Delta t A$$

  is the SPD Helmoltz matrix associated with implicit treatment of the diffusion term.

- **Q:** What is the maximum timestep size, Δt, that we can use?

SEM in 2D and 3D

SEM in 2D and 3D

- Objectives:
  - Look at function definitions in 2D for a single element.
  - Evaluate the Laplace operator $\underline{w} := A\underline{u}$ in 2D and 3D.
  - Explore *preconditioning* strategies for *iterative solution* of $A\underline{u} = \underline{g}.$
  - Consider convection issues in 2D and 3D.

SEM in Higher Dimensions

$$\text{Poisson:} \quad -\nabla^2 \tilde{u} = f, \longrightarrow -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = f(x,y), \qquad u|_{\partial\Omega} = 0.$$

$$\text{WRT:} \quad -\int_\Omega v\, \nabla^2 u\, dV = \int_\Omega v\, f\, dV \qquad \forall\, v \in X_0^N \subset \mathcal{H}_0^1.$$

- Integrate by parts —

$$\int_\Omega v\, \nabla^2 u\, dV = \int_\Omega \nabla v \cdot \nabla u\, dV - \int_{\partial\Omega} v\, \cancelto{}{\nabla u} \cdot \mathbf{n}\, dS.$$

- Define a-inner product:

$$a(v,u) := \int_\Omega \nabla v \cdot \nabla u\, dV = \int_\Omega \frac{\partial v}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial u}{\partial y}\, dV$$

SEM in Higher Dimensions

- Define a-inner product:

$$a(v,u) \;:=\; \int_\Omega \nabla v \cdot \nabla u \, dV \;=\; \int_\Omega \frac{\partial v}{\partial x}\frac{\partial u}{\partial x} \;+\; \frac{\partial v}{\partial y}\frac{\partial u}{\partial y}\, dV$$

- Final form: *Find* $u \in X_0^N \subset \mathcal{H}_0^1$ *such that*

$$\boxed{a(v,u) \;=\; (v,f) \qquad \forall\, v \,\in X_0^N.}$$

- Remainder is to evaluate the integrals on the left and right.

- We begin by defining our *basis functions* for a single element (for now).

Spectral Element Basis Functions in 2D

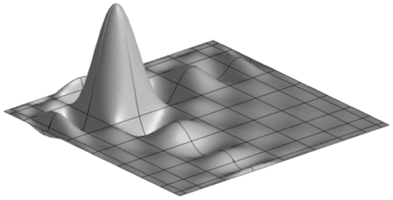
- Nodal (Lagrangian) basis:

$$u(x,y)|_{\Omega^e} = \sum_{i=0}^{N}\sum_{j=0}^{N} u^e_{ij}\, h_i(r)\, h_j(s)$$

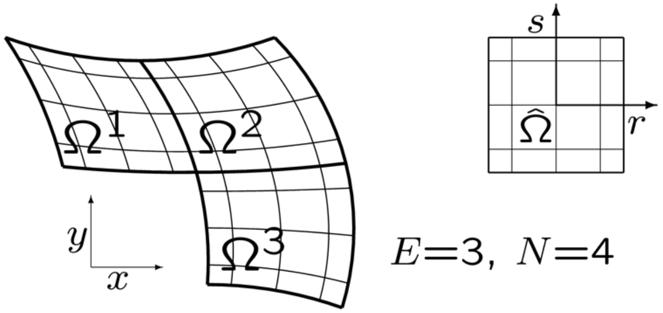
$$h_i(r) \in \mathcal{P}_N(r), \qquad h_i(\xi_j) = \delta_{ij}$$

- ξ_j = Gauss-Lobatto-Legendre quadrature points:
  - stability (*not* uniformly distributed points)
  - allows pointwise quadrature (for *most* operators…)
  - easy to implement BCs and C^0 continuity

- ***Tensor-product forms:*** **key to efficiency!**



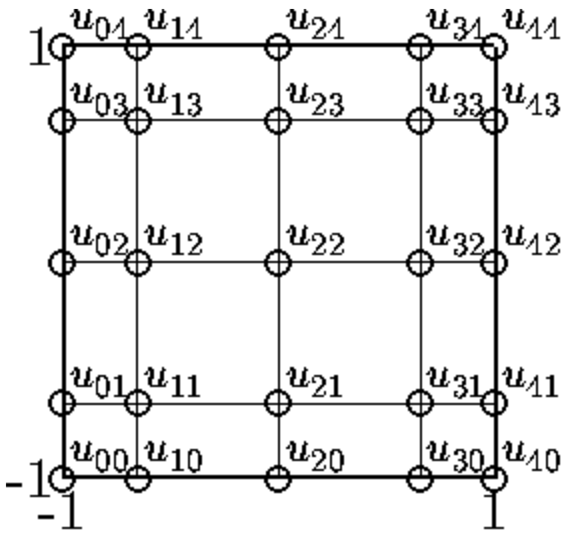
2D basis function, N=10

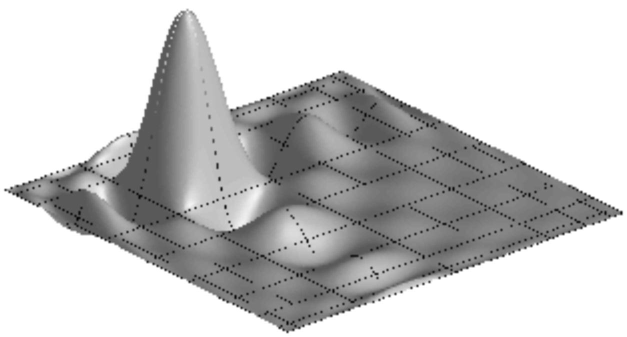


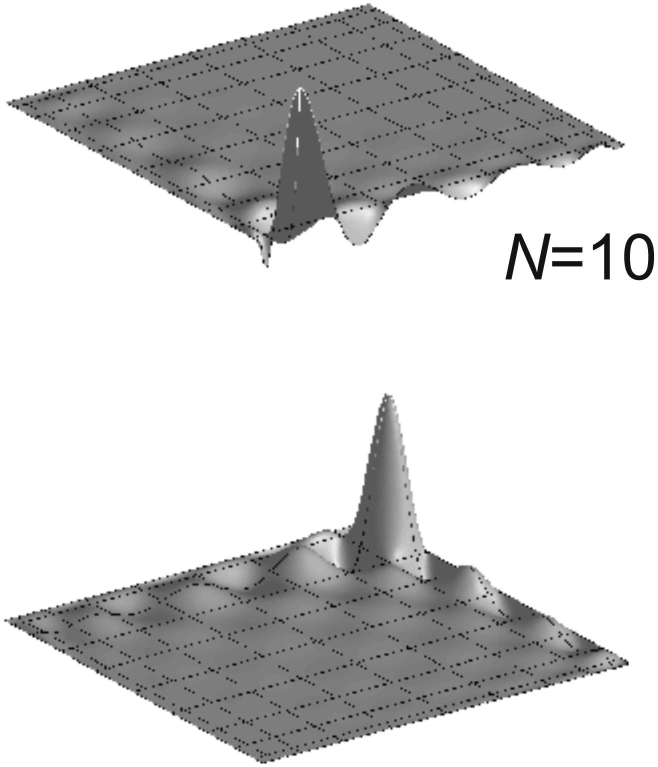
E=3, N=4

Local Spectral Element Basis in 2D

N=4


u_{04} u_{14} u_{24} u_{34} u_{44}
u_{03} u_{13} u_{23} u_{33} u_{43}
u_{02} u_{12} u_{22} u_{32} u_{42}
u_{01} u_{11} u_{21} u_{31} u_{41}
u_{00} u_{10} u_{20} u_{30} u_{40}
1
-1
-1
1

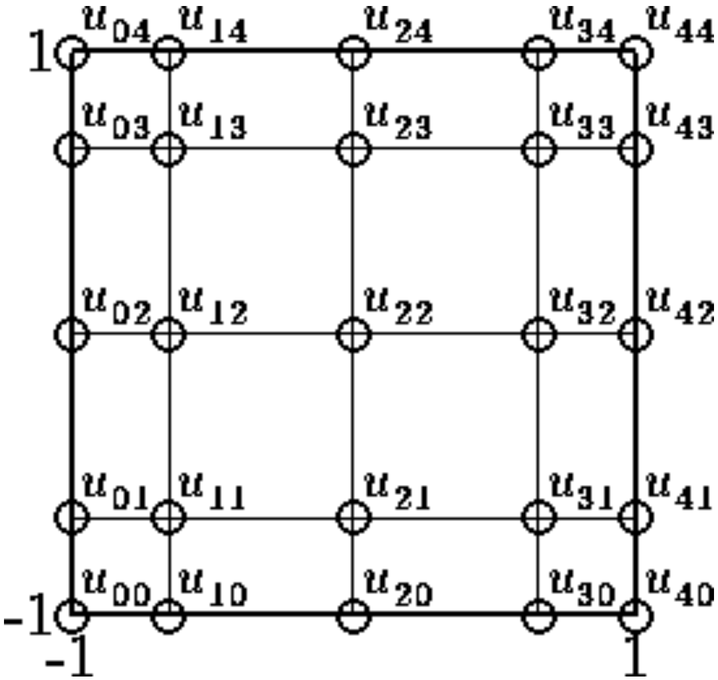




N=10

Spectral Element Operator Evaluation

Consider evaluation of the partial derivative $w_{pq} = \left.\frac{\partial u}{\partial x}\right|_{\xi_p \xi_q}$

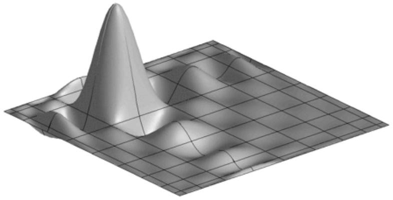


u_{04} u_{14} u_{24} u_{34} u_{44}
u_{03} u_{13} u_{23} u_{33} u_{43}
u_{02} u_{12} u_{22} u_{32} u_{42}
u_{01} u_{11} u_{21} u_{31} u_{41}
u_{00} u_{10} u_{20} u_{30} u_{40}
1
-1
-1
1

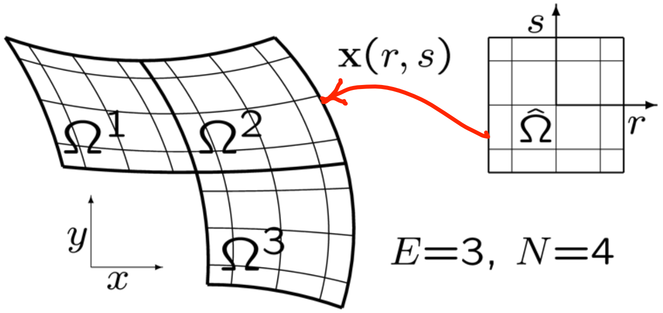
$$\begin{pmatrix} w_{00} \\ w_{10} \\ \vdots \\ w_{N0} \\ w_{01} \\ w_{11} \\ \vdots \\ w_{N1} \\ \vdots \\ w_{0N} \\ w_{1N} \\ \vdots \\ w_{NN} \end{pmatrix} = \begin{pmatrix} \hat{D} & & & & \\ & \hat{D} & & & \\ & & \hat{D} & & \\ & & & \hat{D} & \\ & & & & \hat{D} \end{pmatrix} \begin{pmatrix} u_{00} \\ u_{10} \\ \vdots \\ u_{N0} \\ u_{01} \\ u_{11} \\ \vdots \\ u_{N1} \\ \vdots \\ u_{0N} \\ u_{1N} \\ \vdots \\ u_{NN} \end{pmatrix}$$

$$(u_r)_{ij} = \sum_{k=0}^{N} \hat{D}_{ik} u_{kj}, \qquad (u_s)_{ij} = \sum_{k=0}^{N} \hat{D}_{jk} u_{ik} = \sum_{k=0}^{N} u_{ik} \hat{D}^T_{kj}$$

Geometric Deformation in 2D



2D basis function, N=10



$$\mathbf{x} = (x,y) = (x_1,x_2)$$

$$u(x,y) = \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij}\, h_i(r)\, h_j(s) \;\in\; \mathbb{P}_N(r,s)$$

$$x(r,s) = \sum_{i,j} x_{ij}\, h_i(r)\, h_j(s)\,, \qquad y(r,s) = \sum_{i,j} y_{ij}\, h_i(r)\, h_j(s)\,.$$

- Chain rule:

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial u}{\partial s}\frac{\partial s}{\partial x}\,, \qquad \frac{\partial u}{\partial y} = \frac{\partial u}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial u}{\partial s}\frac{\partial s}{\partial y}$$

$$\frac{\partial v}{\partial x} = \frac{\partial v}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial v}{\partial s}\frac{\partial s}{\partial x}\,, \qquad \frac{\partial v}{\partial y} = \frac{\partial v}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial v}{\partial s}\frac{\partial s}{\partial y}$$

- In $\mathbb{R}^d$:

$$\frac{\partial u}{\partial x_k} = \sum_{k=1}^{d} \frac{\partial u}{\partial r_i}\frac{\partial r_i}{\partial x_k}$$

Evaluation of *a(v,u)*

$$
\begin{aligned}
\mathcal{I} \quad &:= \quad a(v,u) \; = \; \sum_{k=1}^{2} \int_{\Omega} \frac{\partial v}{\partial x_k} \frac{\partial u}{\partial x_k} \, dV \\
&= \quad \sum_{k=1}^{2} \int_{\Omega} \left(\sum_i \frac{\partial v}{\partial r_i} \frac{\partial r_i}{\partial x_k} \right) \left(\sum_j \frac{\partial u}{\partial r_j} \frac{\partial r_i}{\partial x_k} \right) dV \\
&= \quad \sum_i \sum_j \int_{-1}^{1} \int_{-1}^{1} \frac{\partial v}{\partial r_i} \tilde{G}_{ij} \frac{\partial u}{\partial r_j} \, J(r,s) \, dr \, ds \qquad \left\{ \begin{array}{l} \text{apply} \\ \text{quadrature} \end{array} \right. \\
&\approx \quad \sum_i \sum_j \left(\sum_{p=0}^{N} \sum_{q=0}^{N} \rho_p \, \rho_q \; \frac{\partial v}{\partial r_i} \bigg|_{\xi_p, \xi_q} \; (J \tilde{G}_{ij}) \Big|_{\xi_p, \xi_q} \; \frac{\partial u}{\partial r_j} \bigg|_{\xi_p, \xi_q} \right)
\end{aligned}
$$

Here, $\tilde{G}_{ij} \quad := \quad \displaystyle\sum_{k=1}^{d} \frac{\partial r_i}{\partial x_k} \frac{\partial r_j}{\partial x_k} \, ,$

and $J_{pq} = \det \left(\dfrac{\partial x_i}{\partial x_j} \right) = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{vmatrix} = x_r y_s \; - \; x_s y_r \, .$

Evaluation of *a(v,u)*

Now consider the derivatives in the integrand,

$$\left.\frac{\partial u}{\partial r_1}\right|_{\xi_p\xi_q} := \left.\frac{\partial u}{\partial r}\right|_{\xi_p\xi_q} = \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij} \left.\frac{dh_i}{dr}\right|_{\xi_p} h_j(\xi_q)$$

$$= \boxed{\sum_{i=0}^{N} \hat{D}_{pi}\, u_{ij} \;=:\; D_r \underline{u}\,.}$$

$$\left.\frac{\partial u}{\partial r_2}\right|_{\xi_p\xi_q} := \left.\frac{\partial u}{\partial s}\right|_{\xi_p\xi_q} = \boxed{\sum_{j=0}^{N} \hat{D}_{qj}\, u_{ij} \;=:\; D_s \underline{u}\,.}$$

We will insert these, along with $D_r\underline{v}$ and $D_s\underline{v}$ into

$$\mathcal{I} \;\approx\; \sum_i \sum_j \left(\sum_{p=0}^{N}\sum_{q=0}^{N} \rho_p\,\rho_q \left.\frac{\partial v}{\partial r_i}\right|_{\xi_p,\xi_q} \left.(J\tilde{G}_{ij})\right|_{\xi_p,\xi_q} \left.\frac{\partial u}{\partial r_j}\right|_{\xi_p,\xi_q} \right)$$

Evaluation of *a(v,u)*

With a bit of rearranging,

$$
\begin{aligned}
\mathcal{I} \approx a_N(v,u) &= \left(\begin{array}{c} D_r\underline{v} \\ D_s\underline{v} \end{array}\right)^T \left[\begin{array}{cc} G_{11} & G_{12} \\ G_{12} & G_{22} \end{array}\right] \left(\begin{array}{c} D_r\underline{u} \\ D_s\underline{u} \end{array}\right) \\
&= \underline{v}^T \left(\begin{array}{c} D_r \\ D_s \end{array}\right)^T \left[\begin{array}{cc} G_{11} & G_{12} \\ G_{12} & G_{22} \end{array}\right] \left(\begin{array}{c} D_r \\ D_s \end{array}\right) \underline{u} \\
&= \underline{v}^T A \underline{u}
\end{aligned}
$$

- Technically, this is $\bar{A}$, because we've yet to apply the BCs.

- Note the extensive use of *quadrature*, which allows the G_{ij}s to be *diagonal*:

$$
(G_{ij})_{pq} := \rho_p \rho_q \, J_{pq} \sum_{k=1}^{2} \left(\frac{\partial r_i}{\partial x_k} \frac{\partial r_j}{\partial x_k} \right) .
$$

Evaluation of *a(v,u) in R^3*

It should come as no surprise that $\mathcal{I}$ in 3D is given by,

$$\begin{aligned}
\mathcal{I} \approx a_N(v,u) &= \underline{v}^T \begin{pmatrix} D_r \\ D_s \\ D_t \end{pmatrix}^T \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{12} & G_{22} & G_{23} \\ G_{13} & G_{23} & G_{33} \end{bmatrix} \begin{pmatrix} D_r \\ D_s \\ D_t \end{pmatrix} \underline{u} \\
&= \underline{v}^T A \underline{u},
\end{aligned}$$

with

$$(G_{ij})_{lmn} := \rho_l \rho_m \rho_n \, J_{lmn} \sum_{k=1}^{3} \left(\frac{\partial r_i}{\partial x_k} \frac{\partial r_j}{\partial x_k} \right)_{mnl} .$$

- Look at the *memory access costs:* – only $7(N+1)^3$ to evaluate $A\underline{u}$.

- However, if we *store* A, the cost is $(N+1)^6$! (per element!)

- Recall, there are now $(N+1)^3$ unknowns in $\underline{u}$, or in $\underline{u}^e$ in the multi-element case.

Comparison of A in 2D and 1D

Let's compare 2D to 1D:

$$A_{2D} \;=\; \left(\begin{array}{c} D_r \\ D_s \end{array}\right)^T \left[\begin{array}{cc} G_{11} & G_{12} \\ G_{12} & G_{22} \end{array}\right] \left(\begin{array}{c} D_r \\ D_s \end{array}\right),$$

with
$$(G_{ij})_{pq} \;:=\; \rho_p \rho_q \, J_{pq} \sum_{k=1}^{2} \left(\frac{\partial r_i}{\partial x_k} \frac{\partial r_j}{\partial x_k} \right).$$

For 1D,
$$A_{1D} = \hat{D}^T \left(\frac{L}{2} \hat{B} \right) \hat{D} \,.$$

- Here, $\frac{L}{2}$ constitutes the product of the metrics $(\frac{\partial r_i}{\partial x_j})$ and the Jacobian (J), while $\hat{B} := diag(\rho_k)$ accounts for the quadrature weights.

- So, the two have a lot in common, but now we've accomodated geometric flexibility, which is specified through the nodal point distribution (x_{ij}, y_{ij}).

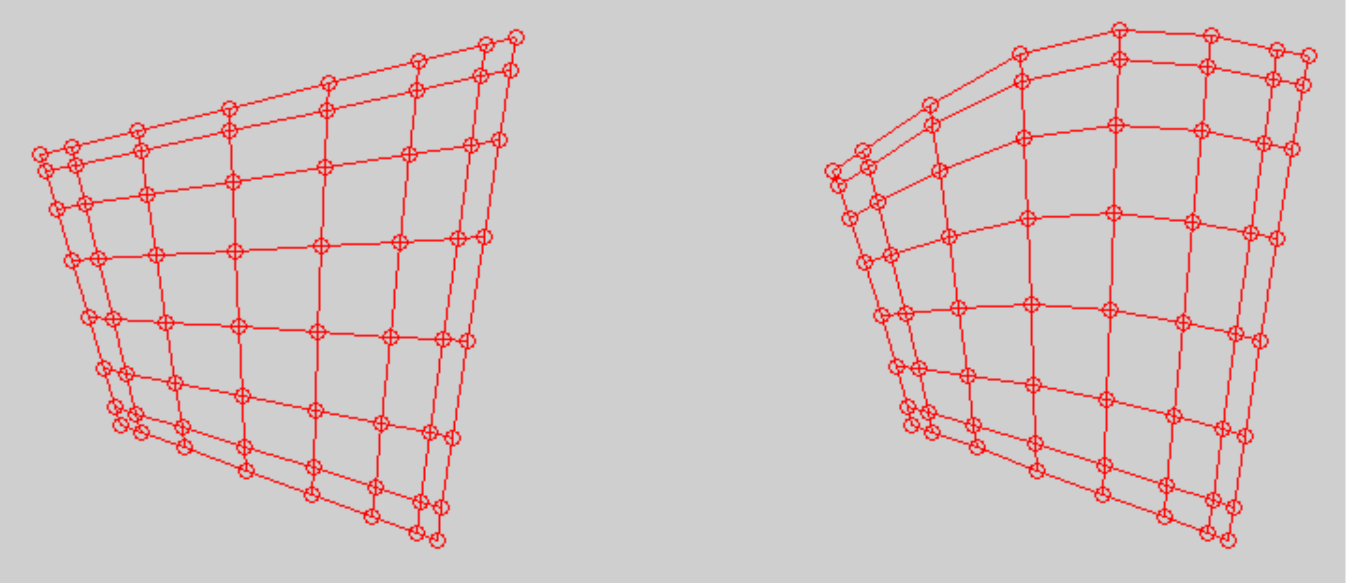
- Moreover, A has the same condition number scaling, $\kappa \sim N^3$ in all space dimensions, $d = 1$, 2, or 3.

Generation of Mesh Deformaton

Gordon-Hall Mapping for Mesh Deformation

- Vertex deformation + Edge perturbations + Face perturbations

- Each perturbation function vanishes at the edge or face boundary, and is blended linearly to the opposite side



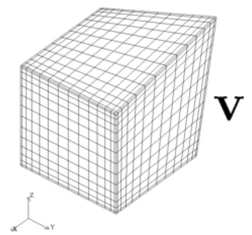
Gordon-Hall Mapping for Mesh Deformation

- Vertex deformation

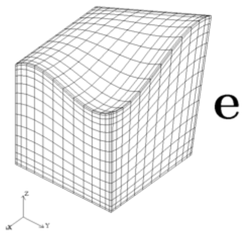
  + Edge perturbations

  + Face perturbations

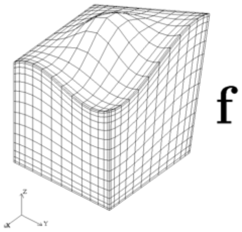
$$\mathbf{v}_{ijk} = \textstyle\sum_{\hat\imath\hat\jmath\hat k} h^1_{\hat\imath}(\xi^N_i) h^1_{\hat\jmath}(\xi^N_j) h^1_{\hat k}(\xi^N_k) \tilde{\mathbf{x}}_{\hat\imath N,\hat\jmath N,\hat k N}$$



$$\begin{aligned}\mathbf{e}_{ijk} = \mathbf{v}_{ijk} &+ \textstyle\sum_{\hat\jmath\hat k} h^1_{\hat\jmath}(\xi^N_j) h^1_{\hat k}(\xi^N_k)(\tilde{\mathbf{x}}_{i,\hat\jmath N,\hat k N} - \mathbf{v}_{i,\hat\jmath N,\hat k N}) \\ &+ \textstyle\sum_{\hat\imath\hat k} h^1_{\hat\imath}(\xi^N_i) h^1_{\hat k}(\xi^N_k)(\tilde{\mathbf{x}}_{\hat\imath N,j,\hat k N} - \mathbf{v}_{\hat\imath N,j,\hat k N}) \\ &+ \textstyle\sum_{\hat\imath\hat\jmath} h^1_{\hat\imath}(\xi^N_i) h^1_{\hat\jmath}(\xi^N_j)(\tilde{\mathbf{x}}_{\hat\imath N,\hat\jmath N,k} - \mathbf{v}_{\hat\imath N,\hat\jmath N,k})\end{aligned}$$



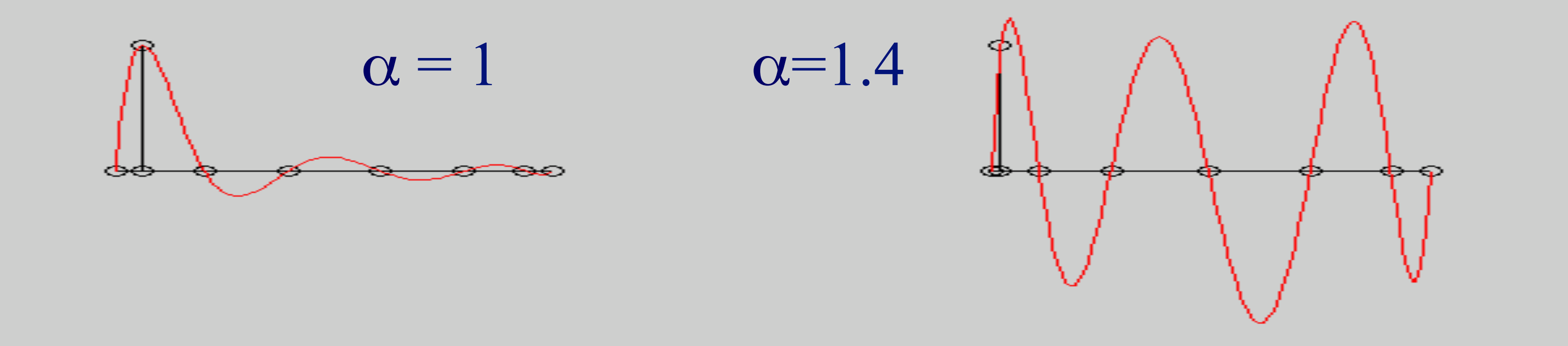
$$\begin{aligned}\mathbf{f}_{ijk} = \mathbf{e}_{ijk} &+ \textstyle\sum_{\hat\imath} h^1_{\hat\imath}(\xi^N_i)(\tilde{\mathbf{x}}_{\hat\imath N,j,k} - \mathbf{e}_{\hat\imath N,j,k}) \\ &+ \textstyle\sum_{\hat\jmath} h^1_{\hat\jmath}(\xi^N_j)(\tilde{\mathbf{x}}_{i,\hat\jmath N,k} - \mathbf{e}_{i,\hat\jmath N,k}) \\ &+ \textstyle\sum_{\hat k} h^1_{\hat k}(\xi^N_k)(\tilde{\mathbf{x}}_{i,j,\hat k N} - \mathbf{e}_{i,j,\hat k N})\end{aligned}$$



Care In Mesh Morphing

- Mesh morphing is very easy and adequate for many applications.

- Care must be used with non-affine mappings. Otherwise, the stability derived from the GLL point distribution may be lost, e.g., stretching $x=r^{\alpha}$:



Can be cured by first morphing entire mesh, extracting vertex values, and re-applying Gordon-Hall (in Nek5000, usrdat() instead of usrdat2())

- Must avoid vertex angles near 0 and 180 deg – ill-conditioned systems.

Impact of Mesh on Iteration Convergence

- Iteration performance for conjugate-gradient iteration w/ overlapping Schwarz preconditioning
- For "shape-regular" elements, iteration count is bounded w.r.t. E & N.

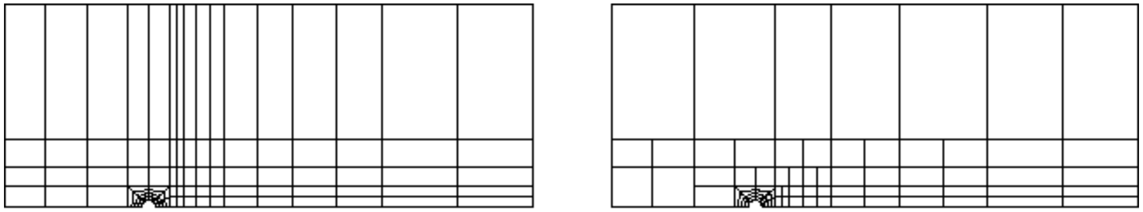
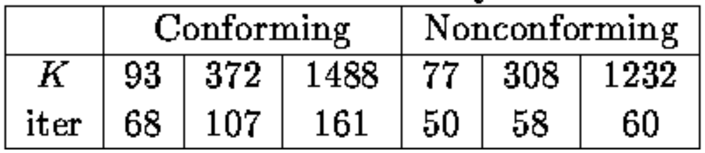


Figure 1: K=93 conforming (left) and K = 77 nonconforming (right) spectral element meshes for flow past a cylinder.

Table 1: Iteration Count for Cylinder Problem



	Conforming			Nonconforming		
K	93	372	1488	77	308	1232
iter	68	107	161	50	58	60

Iteration count bounded with refinement - *scalable*

Enforcing Continuity in 2D

- Recall our matrix assembly in 1D, which is the same in 2D:

$$\mathcal{I} \;=\; \begin{pmatrix} \underline{v}^1 \\ \underline{v}^2 \\ \vdots \\ \underline{v}^e \\ \vdots \\ \underline{v}^E \end{pmatrix}^T \begin{pmatrix} A^1 & & & & & \\ & A^2 & & & & \\ & & \ddots & & & \\ & & & A^e & & \\ & & & & \ddots & \\ & & & & & A^E \end{pmatrix} \begin{pmatrix} \underline{u}^1 \\ \underline{u}^2 \\ \vdots \\ \underline{u}^e \\ \vdots \\ \underline{u}^E \end{pmatrix}$$

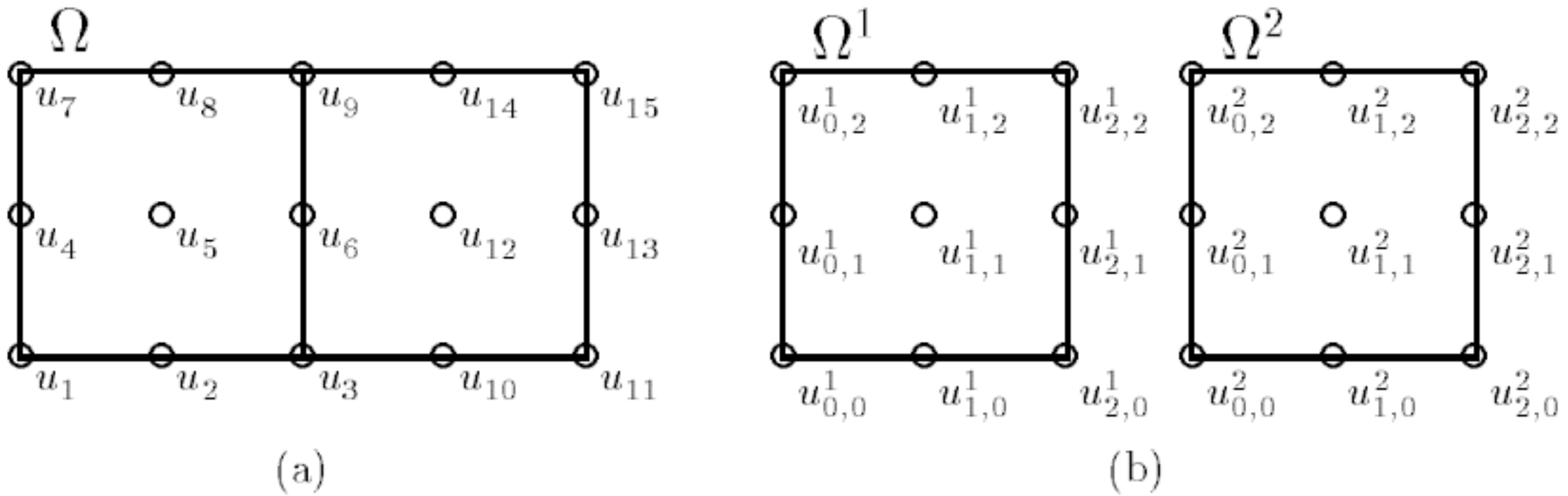
$$= \;\; \underline{v}_L^T A_L \, \underline{u}_L \;=\; \underline{v}^T Q^T A_L \, Q \underline{u} \;=\; \underline{v}^T A \underline{u}$$

- To compute the matrix-vector product $A\underline{u}$ without assembly, we need to effect the action of Q and Q^T.

- This is typically done via subroutines, e.g., as in the following example.

Enforcing Continuity in 2D

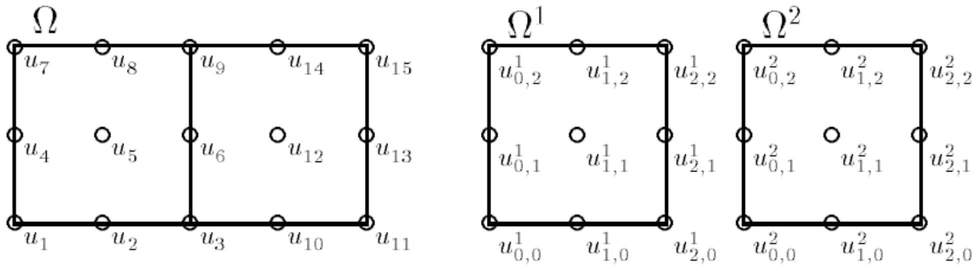
- Consider the following example:



Enforcing Continuity in 2D

- The corresponding Q matrix is:

$$
\text{(c)}\quad
\underbrace{\left(\begin{array}{c}
u^1_{0,0}\\ u^1_{1,0}\\ u^1_{2,0}\\ \hline
u^1_{0,1}\\ u^1_{1,1}\\ u^1_{2,1}\\ \hline
u^1_{0,2}\\ u^1_{1,2}\\ u^1_{2,2}\\ \hline\hline
u^2_{0,0}\\ u^2_{1,0}\\ u^2_{2,0}\\ \hline
u^2_{0,1}\\ u^2_{1,1}\\ u^2_{2,1}\\ \hline
u^2_{0,2}\\ u^2_{1,2}\\ u^2_{2,2}
\end{array}\right)}_{u_L}
=
\underbrace{\left[\begin{array}{ccc|ccc|ccc||cc|cc|cc}
1&&&&&&&&&&&&&&\\
&1&&&&&&&&&&&&&\\
&&1&&&&&&&&&&&&\\ \hline
&&&1&&&&&&&&&&&\\
&&&&1&&&&&&&&&&\\
&&&&&1&&&&&&&&&\\ \hline
&&&&&&1&&&&&&&&\\
&&&&&&&1&&&&&&&\\
&&&&&&&&1&&&&&&\\ \hline\hline
&&1&&&&&&&&&&&&\\
&&&&&&&&&1&&&&&\\
&&&&&&&&&&1&&&&\\ \hline
&&&&&1&&&&&&&&&\\
&&&&&&&&&&&1&&&\\
&&&&&&&&&&&&1&&\\ \hline
&&&&&&&&1&&&&&&\\
&&&&&&&&&&&&&1&\\
&&&&&&&&&&&&&&1
\end{array}\right]}_{Q}
\underbrace{\left(\begin{array}{c}
u_1\\ u_2\\ u_3\\ \hline
u_4\\ u_5\\ u_6\\ \hline
u_7\\ u_8\\ u_9\\ \hline\hline
u_{10}\\ u_{11}\\ \hline
u_{12}\\ u_{13}\\ \hline
u_{14}\\ u_{15}
\end{array}\right)}_{u}
$$



(a) (b)

Q and Q^T implemented as subroutines

In the pseudo-code below, we rely on an array *global_index* that points each local index to its global counterpart.

```
Procedure  $\underline{u}_L = Q\underline{u}$
for  $e = 1, \dots, E$;
for  $j = 0, \dots, N$;
for  $i = 0, \dots, N$;
    $\hat{\imath} := global\_index(i, j, e)$
    $\underline{u}^e_{ij} := \underline{u}_{\hat{\imath}}$
end
```

```
Procedure  $\underline{v} = Q^T \underline{u}_L$
$\underline{v} := \underline{0}$:
for  $e = 1, \dots, E$;
for  $j = 0, \dots, N$;
for  $i = 0, \dots, N$;
    $\hat{\imath} := global\_index(i, j, e)$
    $\underline{v}_{\hat{\imath}} = \underline{v}_{\hat{\imath}} + \underline{u}^e_{ij}$
end
```

- Note that Q^T implies addition.

- In parallel, application of Q and Q^T implies communication. (We discuss this issue off-line, time permitting, but see the reference in *High Order Methods for Incompressible Flow*, Deville, Fischer, Mund, Cambridge, 2002.)

- A scalable ($>$ million-core) stand-alone C code for this gather-scatter operation is provided in the *gs* code within Nek5000.

Fast Operator Evaluation in 2D

- Fast operator evaluation is central to the success of the SEM.

- The end user is interested in a solution to a given accuracy, as fast as possible.

- The rapid convergence of high-order methods (often) implies a need for fewer points. If it takes 10 times, longer to get the result, however, the method is not interesting.

- It turns out—for several reasons—that a properly implemented SEM is competitive with traditional methods on a point-by-point cost basis, which implies lower costs for the SEM because of the reduction in number of points.

- Many of the ideas central to the performance of the SEM were laid out by Steve Orszag in a seminal 1980 JCP article.

- These ideas were an insightful extension of his pioneering work in spectral methods in the 1970s.

Fast Operator Evaluation in 2D

We need to evaluate matrix vector products of the form $\underline{w} = A\underline{u}$:

$$\underline{w} \;=\; \left(\begin{array}{c} D_r \\ D_s \end{array} \right)^T \left[\begin{array}{cc} G_{11} & G_{12} \\ G_{12} & G_{22} \end{array} \right] \left(\begin{array}{c} D_r \\ D_s \end{array} \right) \underline{u}.$$

- We do this one step at a time, starting with $D_r\underline{u}$ and $D_s\underline{u}$.

- To begin, let's recognize that the vector of unknowns, $\underline{u} = (u_{00}\, u_{10}\, \dots\, u_{NN})^T = \{u_{ij}\}$ can also be view as a matrix, $U = u_{ij}$.

- We use this fact to rewrite the matrix-vector product $D_r\underline{u}$ as a *matrix-matrix* product, $\hat{D}U$:

$$D_r\underline{u} := \sum_{k=0}^{N} \hat{D}_{ik} u_{kj} = \hat{D}U.$$

- For the s-derivative, we have a similar result:

$$D_s\underline{u} := \sum_{k=0}^{N} \hat{D}_{jk} u_{ik} = \sum_{k=0}^{N} u_{ik} \hat{D}_{jk} = \sum_{k=0}^{N} u_{ik} \hat{D}_{kj}^T = U\hat{D}^T.$$

- Matrix-matrix products are intrisically fast. *WHY?*

Fast Operator Evaluation in 2D

Using $D_r \underline{u} = \hat{D} U$, $D_s \underline{u} = U \hat{D}^T$, and

$$\underline{w} \quad = \quad \left(\begin{array}{c} D_r \\ D_s \end{array} \right)^T \left[\begin{array}{cc} G_{11} & G_{12} \\ G_{12} & G_{22} \end{array} \right] \left(\begin{array}{c} D_r \\ D_s \end{array} \right) \underline{u},$$

we have the following matlab code for $\underline{w} = A\underline{u}$:

```
ur=Dh*u; us=u*Dh';
t1=G11.*ur + G12.*us;
t2=G12.*ur + G22.*us;
w = Dh'*t1 + t2*Dh;
```

Fast operator evaluation is central to efficient implementation of iterative solvers, which are the fastest possible for 3D problems.

Matlab Demo: mycg.m

```
[Ah,Bh,Ch,Dh,zh,wh]=SEMhat(N);
nb=N+1; R=speye(nb); R=R(2:nb-1,:); R1=R; n1=size(R1,1);
nb=N+1; R=speye(nb); R=R(2:nb  ,:); R2=R; n2=size(R2,1);

% Compute Metrics and Jacobian using Cramer's Rule for 2x2:
[Y,X]=meshgrid(zh,zh);  % Deform X&Y at this point, if you wish...
Y=1*Y;

% Compute Metrics and Jacobian using Cramer's Rule for 2x2:
%    / rx ry \      / xr xs \ -1     1   /  ys -xs \
%    |       |  =   |       |     = ---  |         | ;  J = xr*ys - xs*yr
%    \ sx sy /      \ yr ys /        J   \ -yr  xr /

xr=Dh*X; yr=Dh*Y; xs=X*Dh'; ys=Y*Dh'; J=xr.*ys-xs.*yr;
rx=ys./J; ry=-xs./J; sx=-yr./J; sy=xr./J;

Bb=wh*wh'; %Diagonal mass matrix on ref. domain: B=rho_i rho_j

G11 = Bb.*J.*(rx.*rx + ry.*ry);  % Pointwise collocation
G12 = Bb.*J.*(rx.*sx + ry.*sy);  % for all of these terms!
G22 = Bb.*J.*(sx.*sx + sy.*sy);

fL = 1 + 0*X;                                  % Set rhs:
g  = R1*(Bb.*J.*fL)*R2'; g=reshape(g,n1*n2,1); % Make g a vector for pcg.

asem = @(u,Dh,G11,G12,G22,R1,R2)asem_2d(u,Dh,G11,G12,G22,R1,R2);
M  = speye(n1*n2);  % Identity for PCG

tol = 1.e-10; maxit=400;
```

The code shown here implements conjugate gradients using the general $A\underline{u}_L$ kernel.

Matlab Demo: asem_2d.m

Here is the $A\underline{u}_L$ kernel, which relies on precomputed G_{ij} input.

```
function w = asem_2d(u,Dh,G11,G12,G22,R1,R2);

n1 = size(R1,1); n2 = size(R2,1);

ub = reshape(u,n1,n2);    % Vector to "mesh" form
ub = (R1’*ub)*R2;         % Prolongate to full local coordinates

ur=Dh*ub; us=ub*Dh’;
t1=G11.*ur + G12.*us;
t2=G12.*ur + G22.*us;
w = Dh’*t1 + t2*Dh;

w = (R1*w)*R2’;           % Restrict
w = reshape(w,n1*n2,1);   % Convert back into a vector for pcg
```

Preconditioned Conjugate Gradient Iteration

- Starting with a guess $\underline{x}$, the standard PCG algorithm with M as preconditioner runs as follows:

```
Compute $\underline{r} := \underline{b} - A\underline{x}$, $\underline{z} = M^{-1}\underline{r}$, and $\underline{p} := \underline{z}$,
For $k = 0, 1, \ldots$ until convergence:
    $\underline{w} := A\underline{p}$,                          mat-vec
    $\alpha := (\underline{r}, \underline{z})/(\underline{w}, \underline{p})$,
    $\underline{x}_{j+1} := \underline{x} + \alpha\underline{p}$,
    $\underline{r}_{j+1} := \underline{r} - \alpha\underline{w}$,
    $\underline{z}_{j+1} = M^{-1}\underline{r}_{j+1}$,           preconditioner
    $\beta := (\underline{r}, \underline{z})/(\underline{r}, \underline{z})$,
    $\underline{p} := \underline{z} + \beta\underline{p}$,
End.
```

- The number of iterations for m digits of accuracy scales like $k_{\max} \sim m\kappa^{1/2}$, where κ is the condition number of $M^{-1}A$

- The idea of *preconditioning* is to find a matrix M such that $\kappa \sim 1$ and $\underline{z} = M^{-1}\underline{r}$ is easy to compute.

- There are several strategies for preconditioning the SEM.

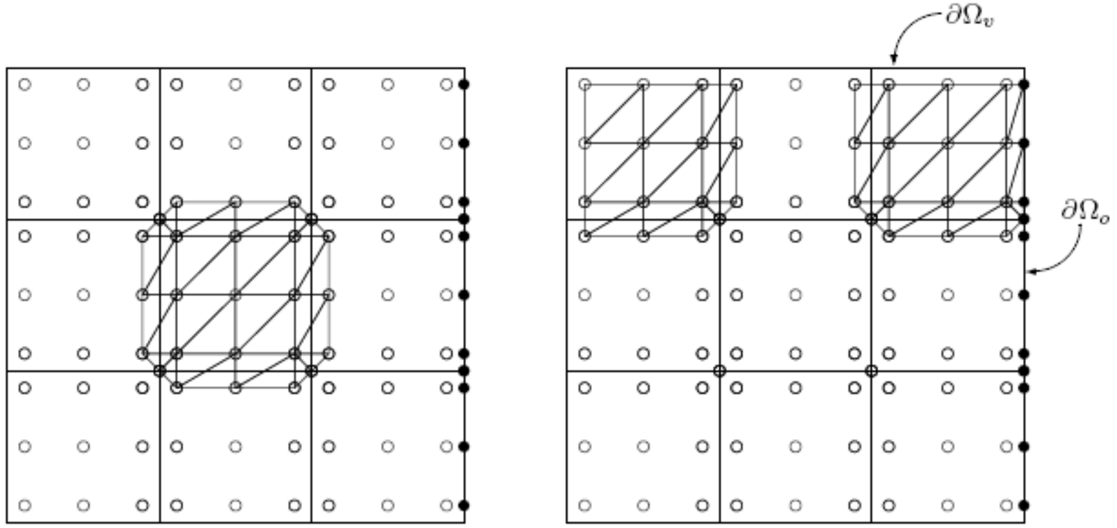
Preconditioned Conjugate Gradient Iteration

- One approach, originally due to Orszag '80, and subsequently explored by Deville & Mund '84 and Canuto & Quarteroni '85, is to set up a *low-order* discretization on the spectral element nodal points.

- Call the resultant—*sparse*—operator A_{fem}.

- The condition number of the preconditioned system, $A_{\text{fem}}^{-1}A$ scales as $\boxed{\kappa \sim \frac{\pi^2}{4},}$ independent of the problem size!

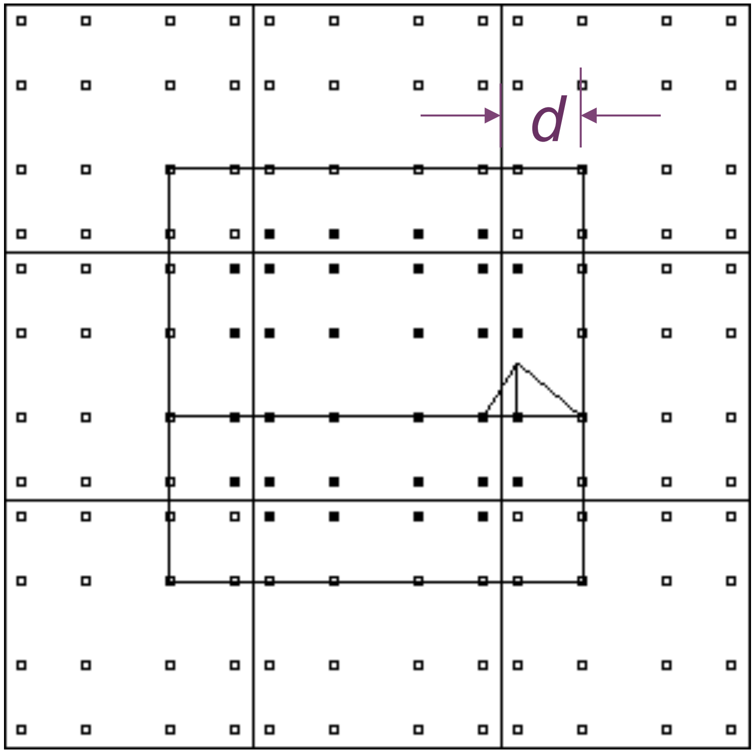
- The advantage here is that the sparse FEM system is much cheaper to solve than the relatively full SEM system. Typically, however, one needs a good algebraic multigrid solver because the resultant FEM mesh has high-aspect ratio cells which are troublesome for most preconditioners.

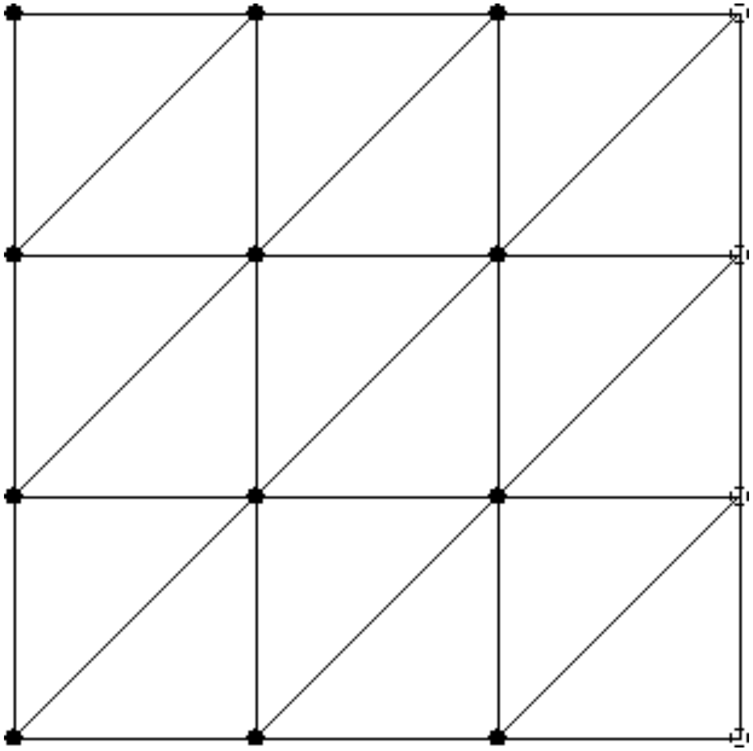


Two-Level Overlapping Additive Schwarz Preconditioner

(Dryja & Widlund 87, Pahl 93, PF 97, FMT 00)

$$\underline{z} = M\underline{r} = \sum_{e=1}^{E} R_e^T A_e^{-1} R_e\, \underline{r} \; + \; R_0^T A_0^{-1} R_0 \underline{r}$$



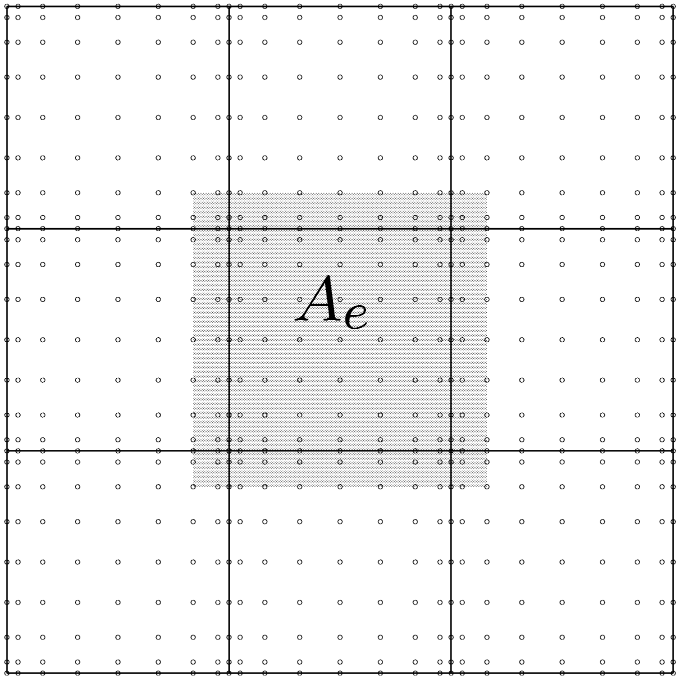


Local Overlapping Solves: FEM-based Poisson problems with homogeneous Dirichlet boundary conditions, A_e .

Coarse Grid Solve: Poisson problem using linear finite elements on entire spectral element mesh, A_0 (GLOBAL).

Overlapping Additive Schwarz Smoother

- ◇ $M_{\text{Schwarz}} = \sum R_e^T A_e^{-1} R_e$ *Dryja & Widlund 87,...*
- ◇ Fast tensor-product solvers for A_e^{-1} *Rice et al. '64, Couzy '95*
- ◇ Bypasses cell aspect-ratio problem



$$A_e^{-1} = (S \otimes S)\,(I \otimes \Lambda_x + \Lambda_y \otimes I)^{-1}\,(S \otimes S)^T$$

Extension to Navier-Stokes

Navier-Stokes Time Advancement

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla p + \tfrac{1}{Re} \nabla^2 \mathbf{u} \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

- Nonlinear term: *explicit*
  - k th-order backward difference formula / extrapolation (k =2 or 3)
  - k th-order characteristics (Pironneau ' 82, MPR '90)
- Linear Stokes problem: pressure/viscous decoupling:
  - 3 Helmholtz solves for velocity *("easy" w/ Jacobi-precond.CG)*
  - (consistent) Poisson equation for pressure *(computationally dominant)*
- For LES, apply grid-scale spectral filter (F. & Mullen 01, Boyd ' 98)
  - – in spirit of HPF model (Schlatter 04)

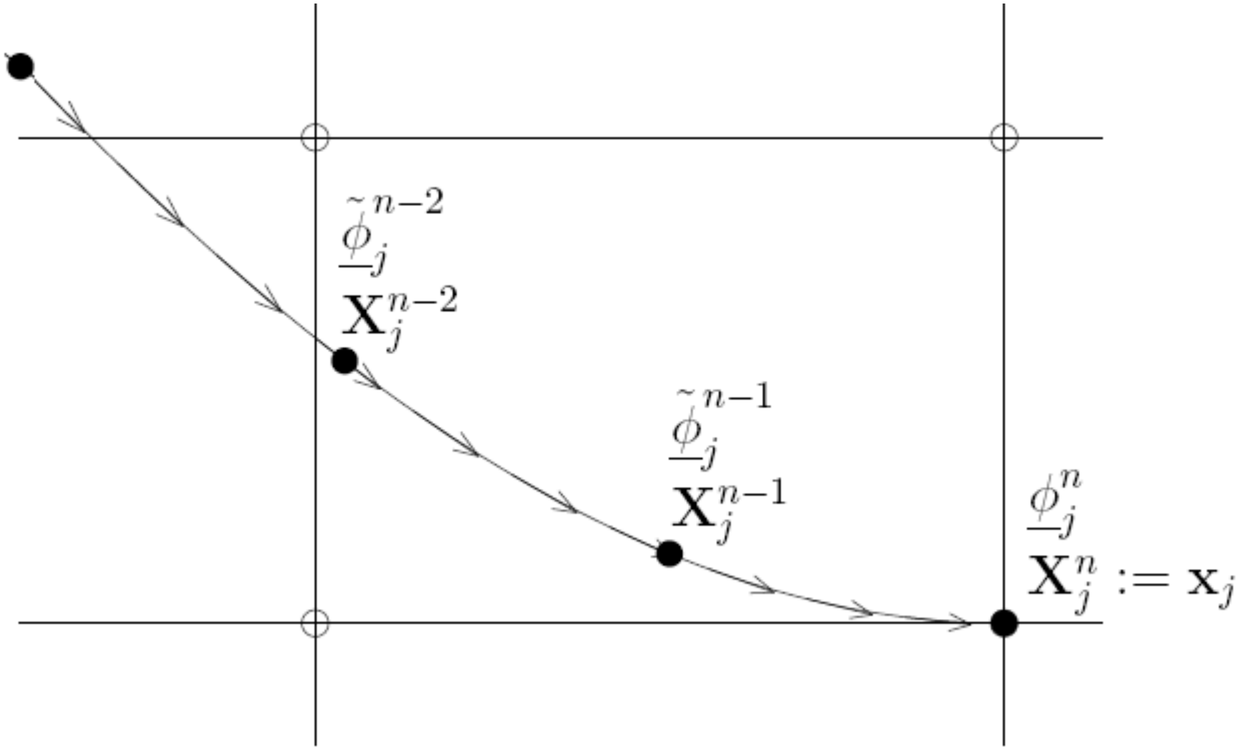
Characteristics-Based Convection Treatment

(OIFS Scheme - Maday, Patera, Ronquist 90, Characteristics - Pironneau 82)

Idea: Solve Navier-Stokes in Lagrangian framework:

$$\frac{D\mathbf{u}}{Dt} = S(\mathbf{u})$$

For a scalar ϕ, we have $\frac{D\phi}{Dt} = \frac{3\phi^n - 4\tilde{\phi}^{n-1} + \tilde{\phi}^{n-2}}{2\Delta t} + O(\Delta t^2)$



$\tilde{\underline{\phi}}_j^{n-2}$
$\mathbf{X}_j^{n-2}$
$\tilde{\underline{\phi}}_j^{n-1}$
$\mathbf{X}_j^{n-1}$
$\underline{\phi}_j^n$
$\mathbf{X}_j^n := \mathbf{x}_j$

Characteristics-Based Convection Treatment

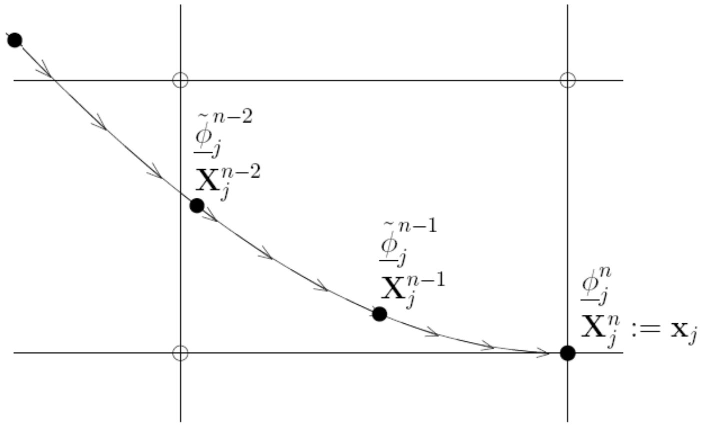
(OIFS Scheme - Maday, Patera, Ronquist 90, Characteristics - Pironneau 82)

For velocity (or ϕ), we compute the values of $\tilde{\mathbf{u}}^{n-q}$ by solving an auxiliary advection problem.

$$\frac{D\mathbf{u}}{Dt} = \frac{3\mathbf{u}^n - 4\tilde{\mathbf{u}}^{n-1} + \tilde{\mathbf{u}}^{n-2}}{2\Delta t} + O(\Delta t^2) = S(\mathbf{u}^n)$$

$$\tilde{\mathbf{u}}^{n-q}: \qquad \frac{\partial \tilde{\mathbf{u}}^{n-q}}{\partial t} + \mathbf{u}\cdot\nabla\tilde{\mathbf{u}}^{n-q} = 0 \quad \text{on } [t^{n-q}, t^n],$$

$$\tilde{\mathbf{u}}^{n-q}(\mathbf{x}, t^{n-q}) := \mathbf{u}^{n-q}(\mathbf{x}, t^{n-q})$$



Unsteady Stokes Problem at Each Step

$$\begin{aligned} \mathcal{H}\,\mathbf{u}^n \;+\; \nabla p^n &= \;\beta_1 \tilde{\mathbf{u}}^{n-1} \;+\; \beta_2 \tilde{\mathbf{u}}^{n-2} \quad \text{in } \Omega\,, \\ \nabla \cdot \mathbf{u}^n \qquad &= \;0 \qquad\qquad\qquad\qquad \text{in } \Omega\,. \end{aligned}$$

$$\mathcal{H} := \left(-\frac{1}{Re} \nabla^2 \;+\; \frac{\beta_0}{\Delta t} \right)$$

$$\beta_0 = \frac{3}{2}, \qquad \beta_1 = 2, \qquad \beta_2 = -\frac{1}{2}$$

- *linear* (allows superposition)
- *implicit* (large CFL, typ. 2-5)
- *symmetric positive definite* operators (conjugate gradient iteration)

P_N - P_{N-2} *Spectral Element Method for Navier-Stokes* (MP 89)

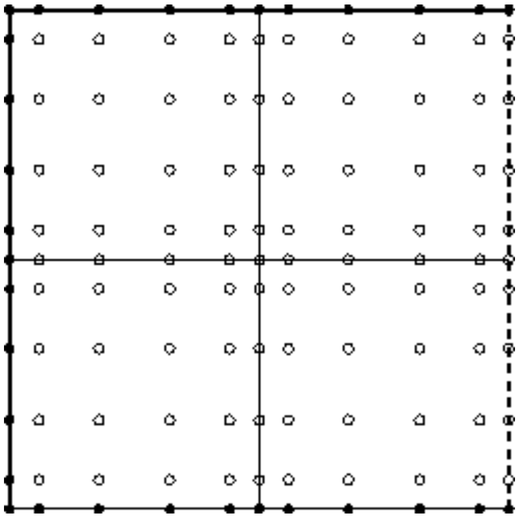
WRT: *Find* $\mathbf{u} \in X^N, p \in Y^N$ *such that:*

$$\frac{1}{Re}(\nabla \mathbf{u}, \nabla \mathbf{v})_{GL} + \frac{1}{\Delta t}(\mathbf{u}, \mathbf{v})_{GL} - (p, \nabla \cdot \mathbf{v})_G = (\mathbf{f}, \mathbf{v})_{GL} \quad \forall\, \mathbf{v} \in X^N \subset H^1$$

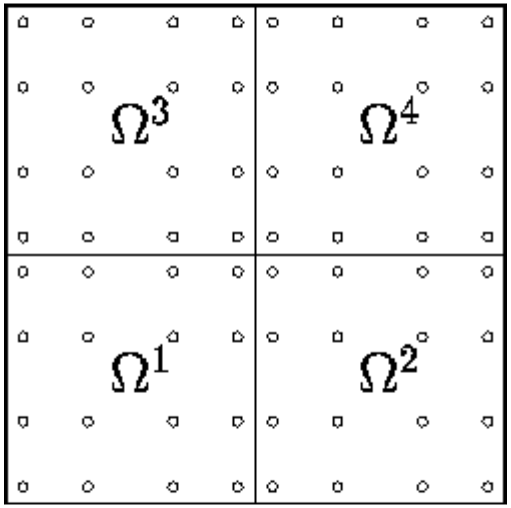
$$- (q, \nabla \cdot \mathbf{u})_G = 0 \qquad \forall\, q \in Y^N \subset L^2$$

Velocity, **u** in P_N, continuous

Pressure, p in P_{N-2}, discontinuous



Gauss-Lobatto Legendre points (velocity)



Gauss Legendre points (pressure)

Navier-Stokes Solution Strategy

- Semi-implicit: explicit treatment of nonlinear term.
- Leads to Stokes saddle problem, which is algebraically split

MPR 90, Blair-Perot 93,

Couzy 95

$$\begin{bmatrix} \mathbf{H} & -\mathbf{D}^T \\ -\mathbf{D} & 0 \end{bmatrix} \begin{pmatrix} \underline{\mathbf{u}}^n \\ \underline{p}^n - \underline{p}^{n-1} \end{pmatrix} = \begin{pmatrix} \mathbf{B}\underline{\mathbf{f}} + \mathbf{D}^T \underline{p}^{n-1} \\ \underline{f}_p \end{pmatrix}$$

$$\begin{bmatrix} \mathbf{H} & -\frac{\Delta t}{\beta_0}\mathbf{H}\mathbf{B}^{-1}\mathbf{D}^T \\ \mathbf{0} & E \end{bmatrix} \begin{pmatrix} \underline{\mathbf{u}}^n \\ \underline{p}^n - \underline{p}^{n-1} \end{pmatrix} = \begin{pmatrix} \mathbf{B}\underline{\mathbf{f}} + \mathbf{D}^T \underline{p}^{n-1} \\ \underline{g} \end{pmatrix} + \begin{pmatrix} \underline{\mathbf{r}} \\ \underline{0} \end{pmatrix} \; . \; ,$$

$$E := \frac{\Delta t}{\beta_0}\mathbf{D}\mathbf{B}^{-1}\mathbf{D}^T \; , \qquad\qquad \underline{\mathbf{r}} = O(\Delta t^2)$$

- ***E*** - consistent Poisson operator for pressure, SPD
  - Stiffest substep in Navier-Stokes time advancement
  - Most compute-intensive phase
  - Spectrally equivalent to SEM Laplacian, ***A***

Pressure Solution Strategy: $\mathbf{E}\underline{\mathbf{p}}^{\mathrm{n}} = \underline{\mathbf{g}}^{\mathrm{n}}$

1. *Projection: compute best approximation from previous time steps*

   – *Compute* $\underline{p}^*$ *in span*$\{\underline{p}^{n\text{-}1}, \underline{p}^{n\text{-}2}, \ldots, \underline{p}^{n\text{-}l}\}$ *through straightforward projection.*

   – *Typically a 2-fold savings in Navier-Stokes solution time.*

   – *Cost: 1 (or 2) matvecs in* E *per timestep*

2. *Preconditioned CG or GMRES to solve*

$$E\,\mathrm{D}\underline{p} = \underline{g}^n - E\,\underline{p}^*$$

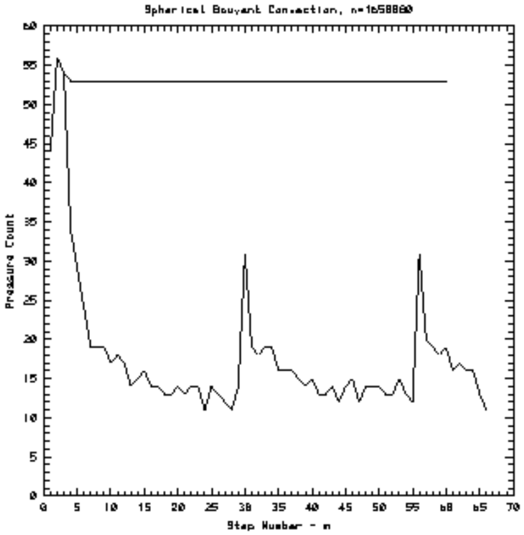
Initial guess for $A\underline{x}^n = \underline{b}^n$ *via projection* (A=E, *SPD)*

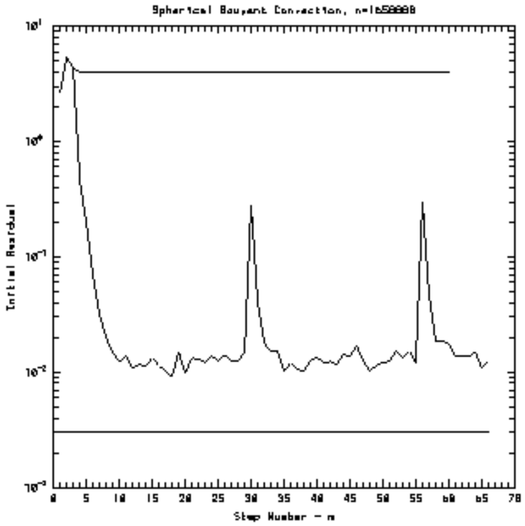
```
Given · $\underline{b}^n$
            · $\{\underline{\tilde{x}}_1, \ldots, \underline{\tilde{x}}_l\}$ satisfying $\underline{\tilde{x}}_i^T A \underline{\tilde{x}}_j = \delta_{ij}$,

· Set $\underline{\bar{x}} := \Sigma\, \alpha_i \underline{\tilde{x}}_i, \quad \alpha_i = \underline{\tilde{x}}_i^T \underline{b}$                (best fit solution)

· Set $\Delta\underline{b} := \underline{b}^n - A\underline{\bar{x}}$

· Solve $A\Delta\underline{x} = \Delta\underline{b}$ to $tol\ \epsilon$                (black box solver)

· $\underline{x}^n := \underline{\bar{x}} + \Delta\underline{x}$

· If $(l = l_{\max})$ then
      $\underline{\tilde{x}}_1 = \underline{x}^n/||\underline{x}^n||_A$
      $l = 1$
  else
      $\underline{\tilde{x}}_{l+1} = (\Delta\underline{x} - \Sigma\,\beta_i\underline{\tilde{x}}_i)/(\Delta\underline{x}^T A\Delta\underline{x} - \Sigma\,\beta_i^2)^{\frac{1}{2}}, \quad \beta_i = \underline{\tilde{x}}_i A\Delta\underline{x}$
      $l = l + 1$
  endif
```

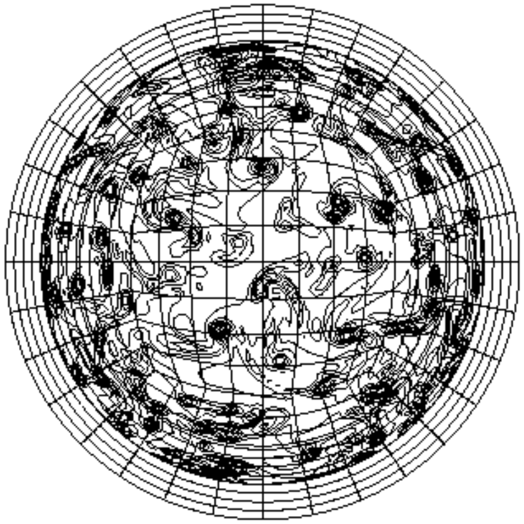
Initial guess for $\mathbf{E}\underline{\mathbf{p}}^{\mathbf{n}} = \underline{\mathbf{g}}^{\mathbf{n}}$ *via projection onto previous solutions*

(F 93 98)

- $\| \underline{p}^n - \underline{p}^* \|_A = O(\mathrm{D}t^l) + O(\, \mathrm{e}_{\mathrm{tol}}\,)$
- two additional mat-vecs per step
- storage: $2 + l_{max}$ vectors
- results with/without projection (1.6 million pressure nodes)



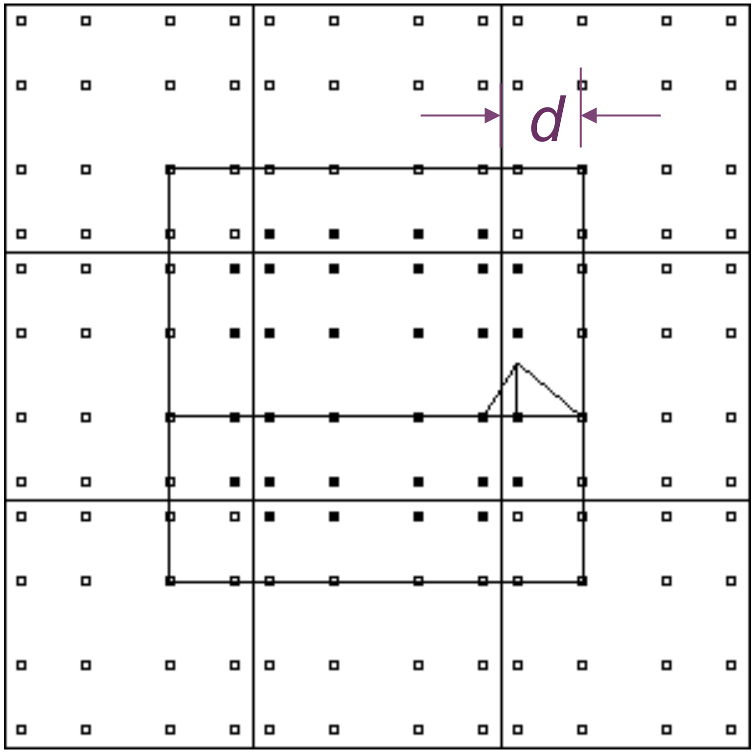




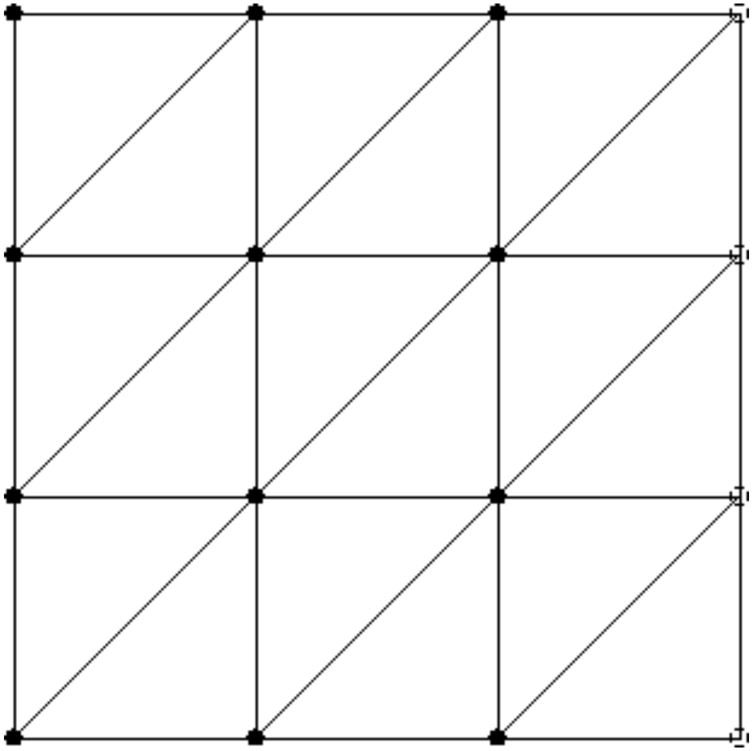
- 4 fold reduction in iteration count, 2 – 4 in typical applications

Overlapping Additive Schwarz Preconditioner for the Pressure

(Dryja & Widlund 87, Pahl 93, PF 97, FMT 00)



***Overlapping Solves**: Poisson problems with homogeneous Dirichlet bcs.*



***Coarse Grid Solve**: Poisson problem using linear finite elements on spectral element mesh (GLOBAL).*

Overlapping Schwarz Precondtioning for Pressure

(Dryja & Widlund 87, Pahl 93, PF 97, FMT 00)

$$\underline{z} = P^{-1}\,\underline{r} = R_0^T A_0^{-1} R_0\,\underline{r} + \sum_{e=1}^{E} R_{o,e}^T A_{o,e}^{-1} R_{o,e}\,\underline{r}$$

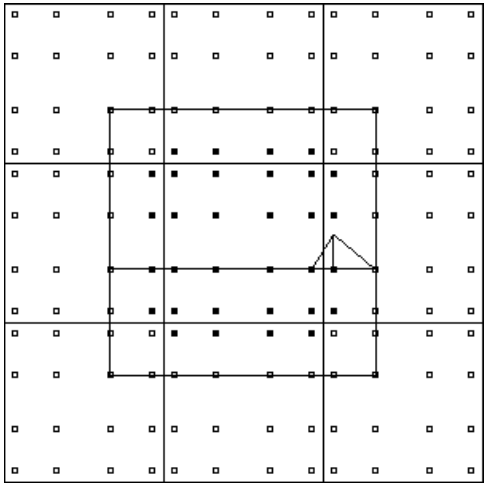
$A_{o,e}$ - low-order FEM Laplacian stiffness matrix on overlapping domain for each spectral element k (Orszag, Canuto & Quarteroni, Deville & Mund, Casarin)

$R_{o,e}$ - Boolean restriction matrix enumerating nodes within overlapping domain e

A_0 - FEM Laplacian stiffness matrix on coarse mesh (~ E x E)

R_0^T - Interpolation matrix from coarse to fine mesh

Overlapping Schwarz - local solve complexity



- Exploit local tensor-product structure
- Fast diagonalization method (FDM) - local solve cost is ~ $4d\,K\,N^{(d+1)}$ (Lynch et al 64)

$$\textbf{2D:}\quad A = (B_y \otimes A_x + A_y \otimes B_x), \qquad S^T A S = \Lambda, \quad S^T B S = I.$$

$$A^{-1} = (S_y \otimes S_x)(I \otimes \Lambda_x + \Lambda_y \otimes I)^{-1}(S_y^T \otimes S_x^T).$$

NOTE: B_x, B_y, *lumped* 1D mass matrices (conditioning)

Op. Count: $W = 8KN^3$ (*vs.* $4KN^3$ for band solve)

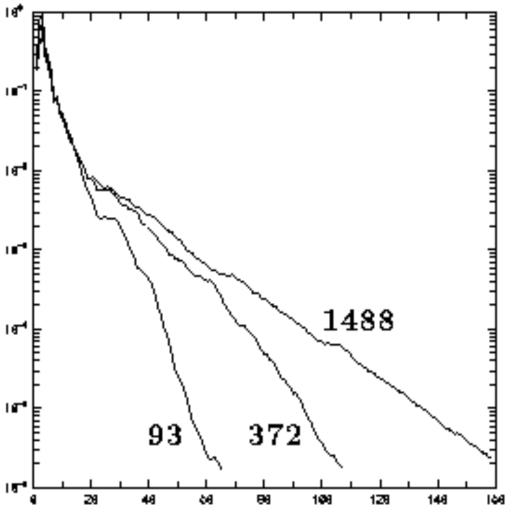
Storage: $S = O(KN^2)$ (*vs.* KN^3 for band solve)

NOTE: $S_y \otimes S_x \underline{u} = S_x U S_y^T$ (matrix-matrix product)

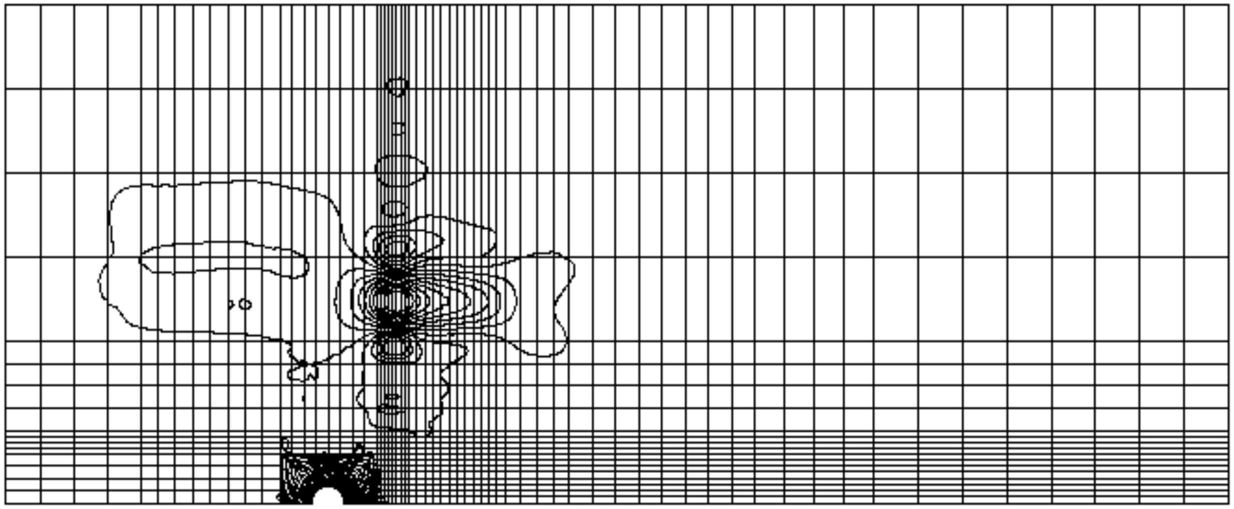
2D Test Problem: Startup flow past a cylinder (N=7)

Performance of the additive Schwarz algorithm, (10^{-5})

	FDM		$N_o = 0$		$N_o = 1$		$N_o = 3$		$A_0 = 0$		Deflation	
K	iter	CPU	iter	CPU	iter	CPU	iter	CPU	iter	CPU	iter	CPU
93	67	4.4	121	10.	64	5.9	49	5.6	169	19.	126	17.
372	114	37.	203	74.	106	43.	73	39.	364	193.	216	125.
1488	166	225.	303	470.	158	274.	107	242.	802	1798.	327	845.



Residual history



Resistant pressure mode, $\underline{p}^{166}$ - $\underline{p}^{25}$, (K=1488)

Impact of High-Aspect Ratio Elements

- Nonconforming discretizations eliminate unnecessary elements in the far field and result in better conditioned systems.

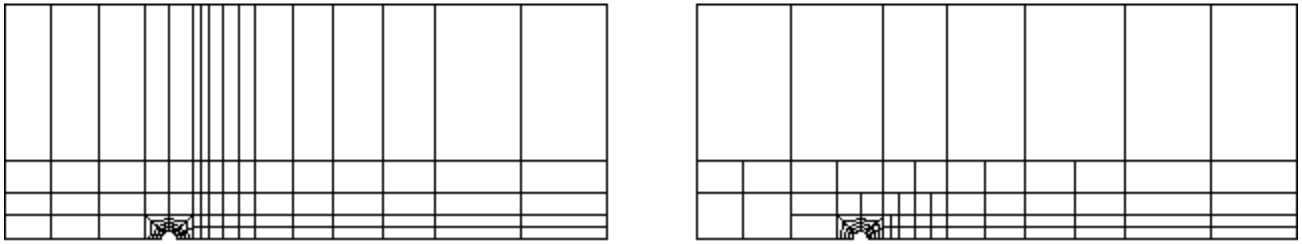


Figure 1: K=93 conforming (left) and K = 77 nonconforming (right) spectral element meshes for flow past a cylinder.

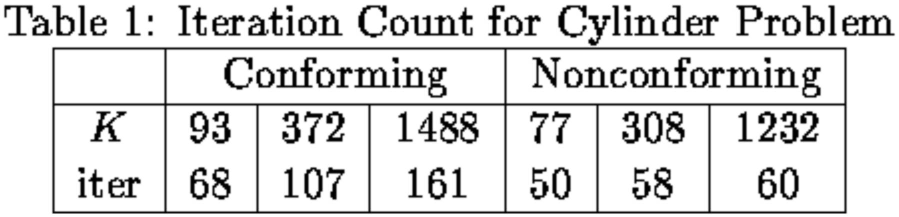


Table 1: Iteration Count for Cylinder Problem

	Conforming			Nonconforming		
K	93	372	1488	77	308	1232
iter	68	107	161	50	58	60

Iteration count bounded with refinement - *scalable*

Stabilizing Convection-Dominated Flows

Stabilizing High-Order Methods

In the absence of eddy viscosity, some type of stabilization is generally required at high Reynolds numbers.

Some options:

- high-order upwinding (e.g., DG, WENO)
- bubble functions
- spectrally vanishing viscosity
- *filtering*
- *dealiasing*

Spectral Filter

Boyd '98, F. & Mullen '01

- Expand in modal basis:

$$u(x) = \sum_{k=0}^{N} \hat{u}_k \, \phi_k(r)$$

- Set filtered function to:

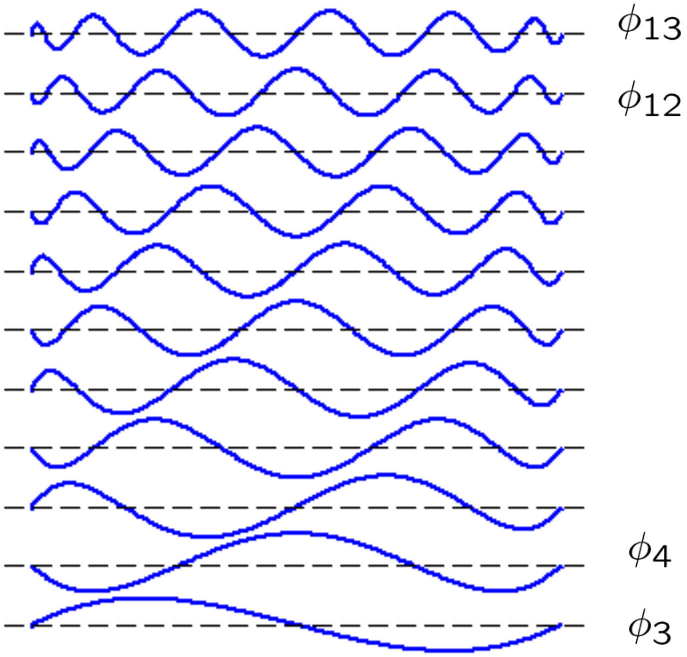
$$\bar{u}(x) = \hat{F}(u) = \sum_{k=0}^{N} \sigma_k \, \hat{u}_k \, \phi_k(r)$$

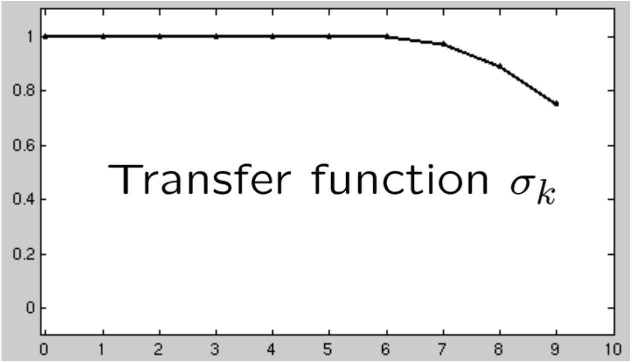
- In higher space dimensions:

$$F = \hat{F} \otimes \hat{F} \otimes \hat{F}$$

- Spectral convergence and continuity preserved. (Coefficients decay exponentially fast.)

- Post-processing (*easy*) !



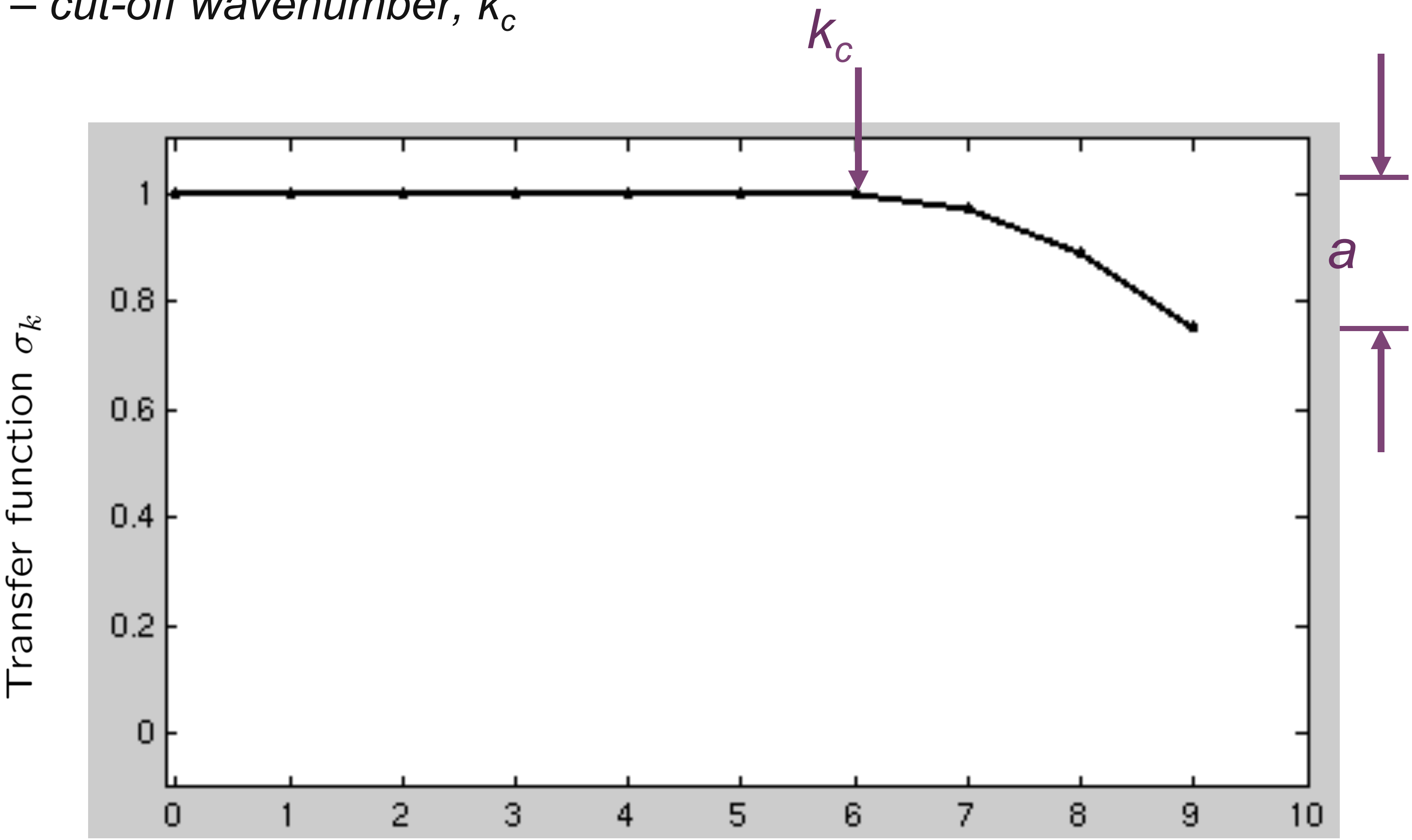


Spectral Filter

Boyd ’98, F. & Mullen ‘01

Transfer function characterized by two parameters:

- *amplitude, a ~ 0.01—0.25*
- *cut-off wavenumber,* k_c



Numerical Stability Test: Shear Layer Roll-Up

(Bell et al. JCP 89, Brown & Minion, JCP 95, F. & Mullen, CRAS 2001)

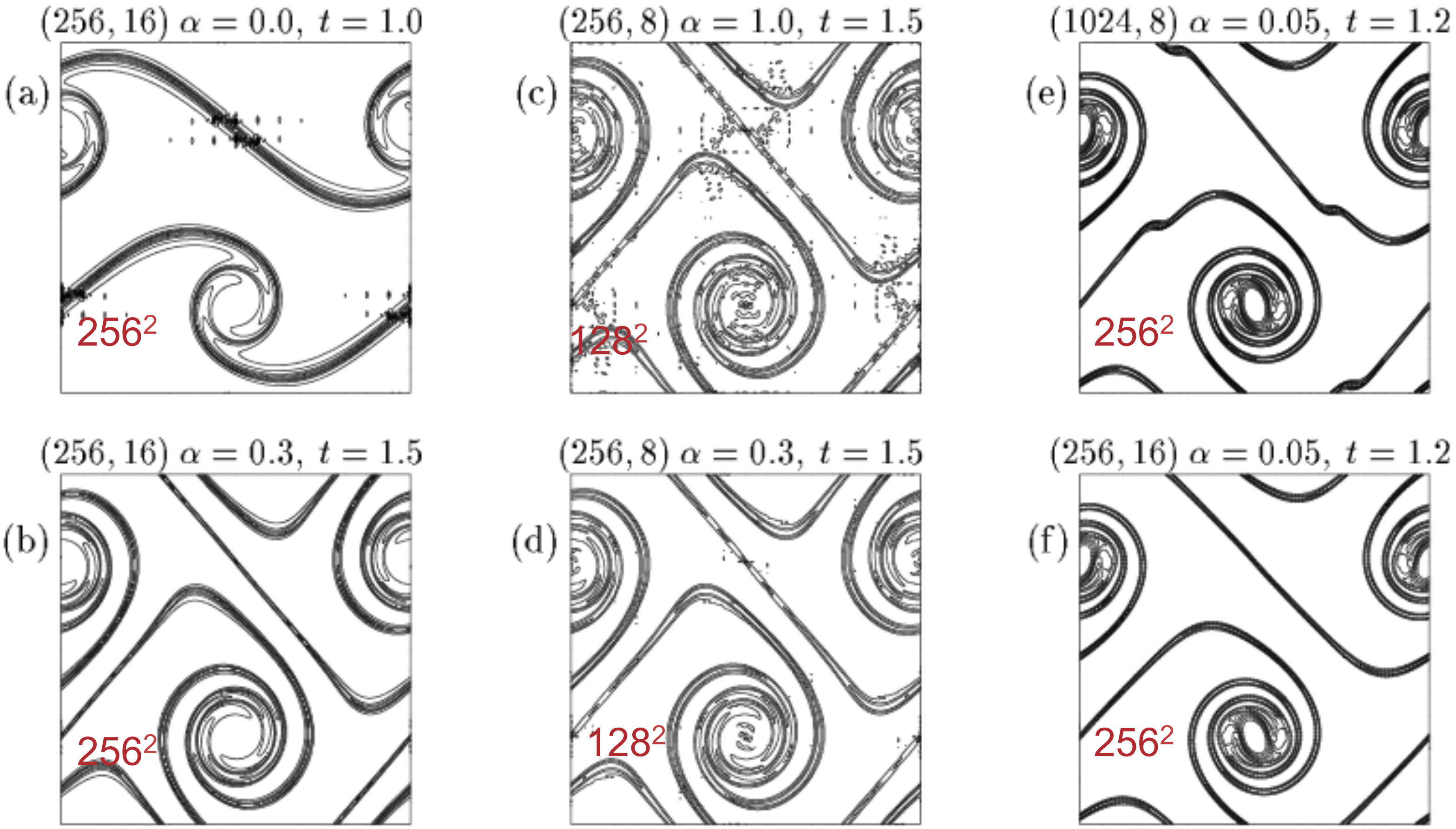


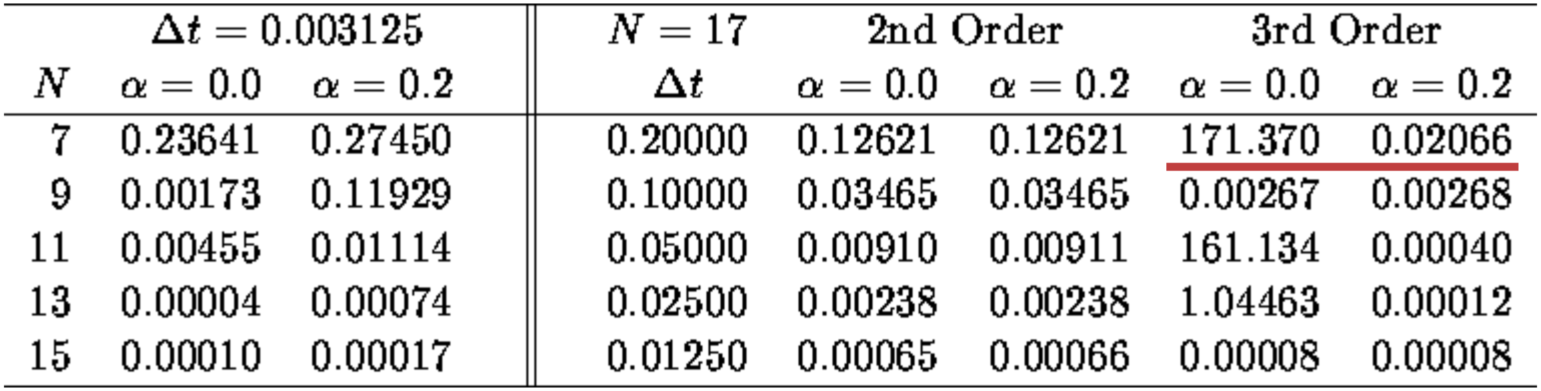
Figure 1: Vorticity for different (K, N) pairings: (a–d) $\rho = 30$, $Re = 10^5$, contours from -70 to 70 by 140/15; (e–f) $\rho = 100$, $Re = 40,000$, contours from -36 to 36 by 72/13. (cf. Fig. 3c in [4]).

Error in Predicted Growth Rate for Orr-Sommerfeld Problem at Re=7500

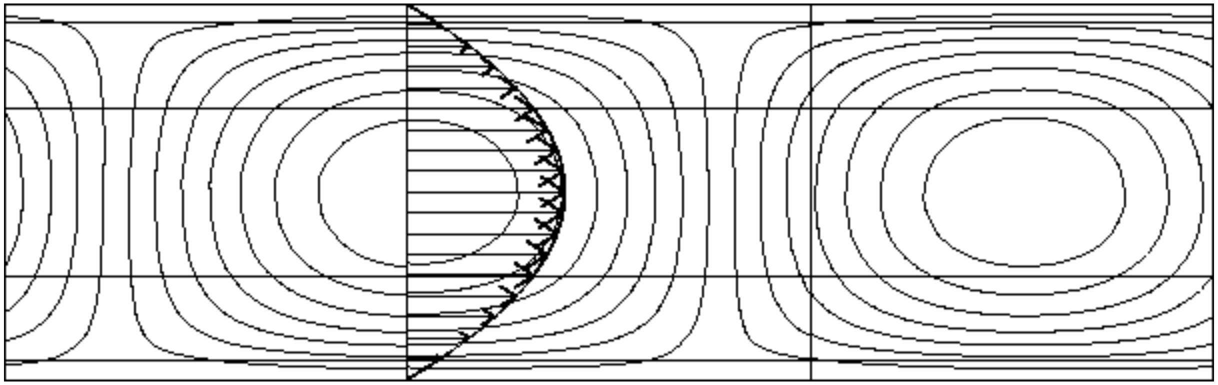
(*Malik & Zang 84*)

Spatial and Temporal Convergence

(F. & Mullen, 01)



	$\Delta t = 0.003125$		$N = 17$	2nd Order		3rd Order	
N	$\alpha = 0.0$	$\alpha = 0.2$	Δt	$\alpha = 0.0$	$\alpha = 0.2$	$\alpha = 0.0$	$\alpha = 0.2$
7	0.23641	0.27450	0.20000	0.12621	0.12621	171.370	0.02066
9	0.00173	0.11929	0.10000	0.03465	0.03465	0.00267	0.00268
11	0.00455	0.01114	0.05000	0.00910	0.00911	161.134	0.00040
13	0.00004	0.00074	0.02500	0.00238	0.00238	1.04463	0.00012
15	0.00010	0.00017	0.01250	0.00065	0.00066	0.00008	0.00008



Base velocity profile and perturbation streamlines

Filtering permits Re$_{d99}$ > 700 for transitional boundary layer calculations

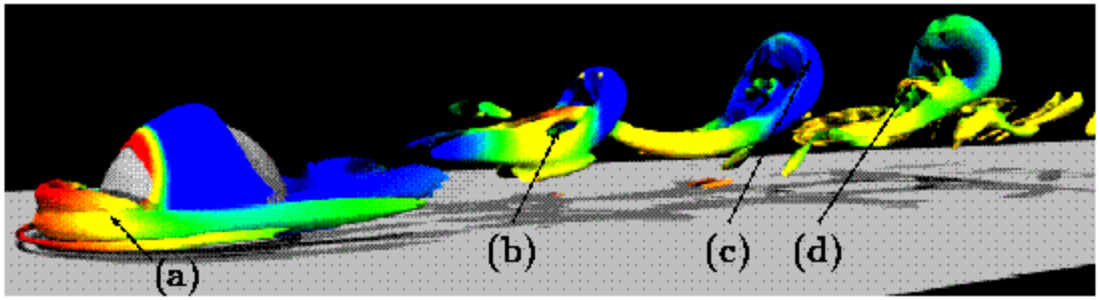
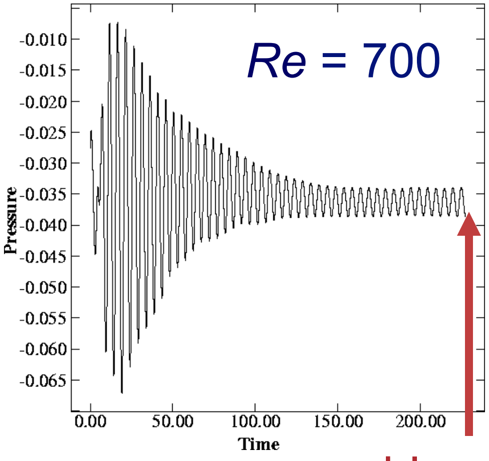
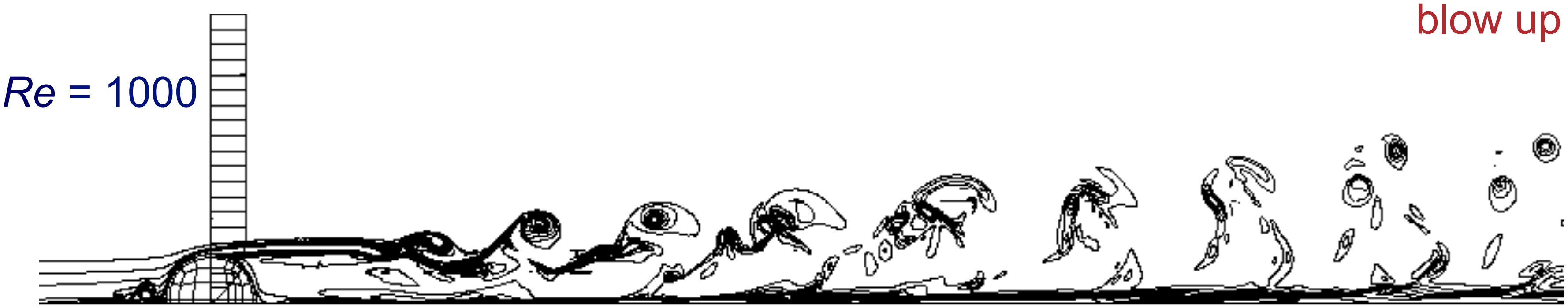
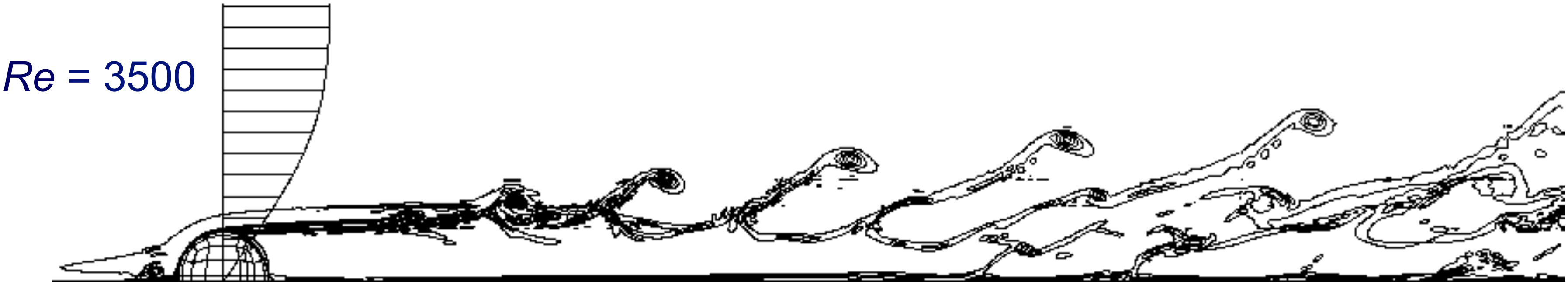


Figure 1: Principal vortex structures identified by $\lambda_2 = -1$ isosurfaces at $Re_k = 760$: standing horseshoe vortex (a), interlaced tails (b), hairpin head (c), and bridge (d). Colors indicate pressure. (K=1021, $N = 15$).



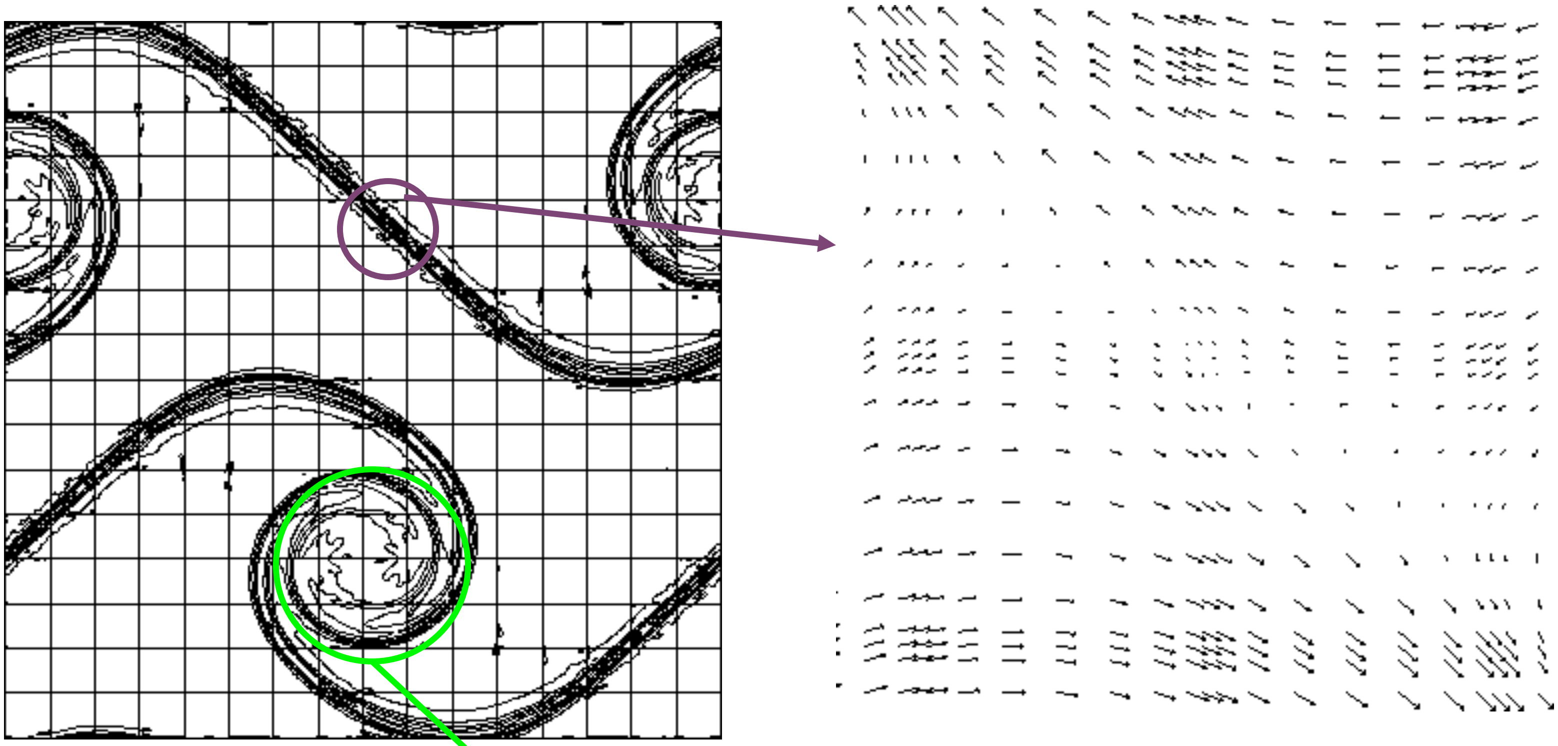
blow up





Why Does Filtering Work ?
(Or, Why Do the Unfiltered Equations Fail? *)*

Double shear layer example:



Ok

High-strain regions are troublesome…

Why Does Filtering Work ?
(Or, Why Do the Unfiltered Equations Fail? *)*

Consider the model problem: $$\frac{\partial u}{\partial t} = -\mathbf{c} \cdot \nabla u$$

Weighted residual formulation: $$B\frac{d\underline{u}}{dt} = -C\underline{u}$$

$$\begin{aligned}
B_{ij} &= \int_\Omega \phi_i \phi_j \, dV = \text{symm. pos. def.} \\
C_{ij} &= \int_\Omega \phi_i \, \mathbf{c} \cdot \nabla \phi_j \, dV \\
&= -\int_\Omega \phi_j \, \mathbf{c} \cdot \nabla \phi_i \, dV - \int_\Omega \phi_j \phi_j \nabla \cdot \mathbf{c} \, dV \\
&= \text{skew symmetric, if } \nabla \cdot \mathbf{c} \equiv 0. \\
B^{-1}C &\longrightarrow \text{imaginary eigenvalues}
\end{aligned}$$

Discrete problem should never blow up.

Why Does Filtering Work ?
(Or, Why Do the Unfiltered Equations Fail? *)*

Weighted residual formulation vs. spectral element method:

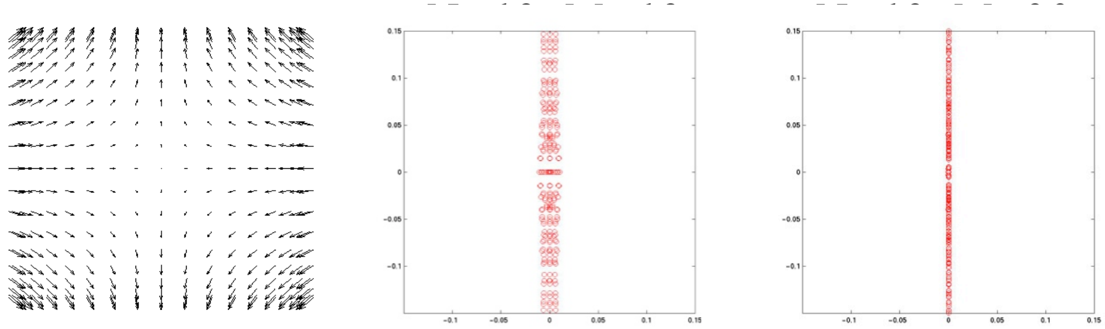
$$\begin{aligned} C_{ij} &= (\phi_i, \mathbf{c} \cdot \nabla \phi_j) = & -C_{ji} \\ \tilde{C}_{ij} &= (\phi_i, \mathbf{c} \cdot \nabla \phi_j)_N \neq & -\tilde{C}_{ji} \end{aligned}$$

This suggests the use of over-integration (dealiasing) to ensure that skew-symmetry is retained

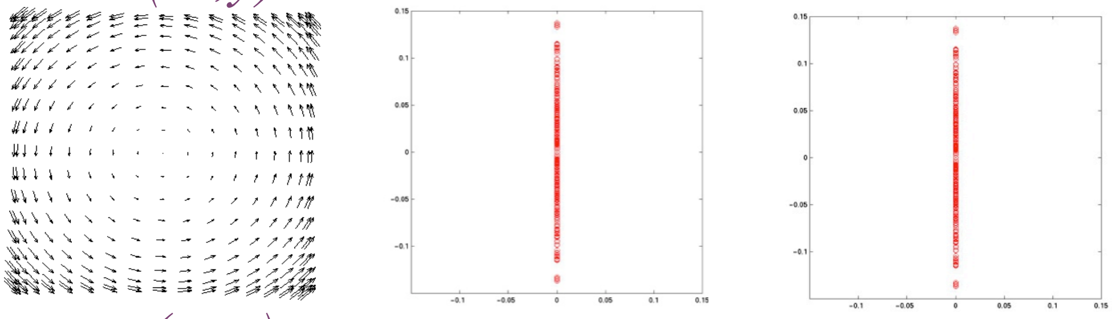
$$\begin{aligned} C_{ij} &= (J\phi_i, (J\mathbf{c}) \cdot J\nabla \phi_j)_M \\ J_{pq} &:= h_q^N(\xi_p^M) \quad \text{interpolation matrix (1D, single element)} \end{aligned}$$

Aliased / Dealiased Eigenvalues: $u_t + \mathbf{c} \cdot \nabla u = 0$

- Velocity fields model first-order terms in expansion of straining and rotating flows.
  - For straining case, $\frac{d}{dt}|u|^2 \sim |\hat{u}_{N\,\cdot}|^2 + |\hat{u}_{\cdot\,N}|^2$
  - Rotational case is skew-symmetric.
  - Filtering attacks the leading-order unstable mode.



c = *(-x,y)*



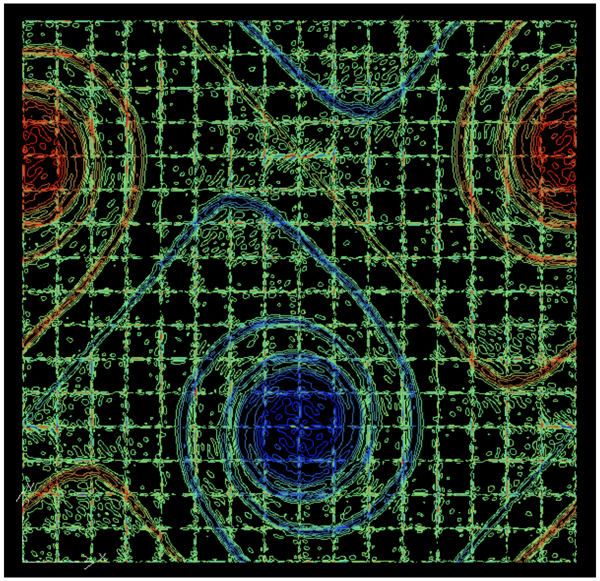
c = *(-y,x)*

Stabilization Summary

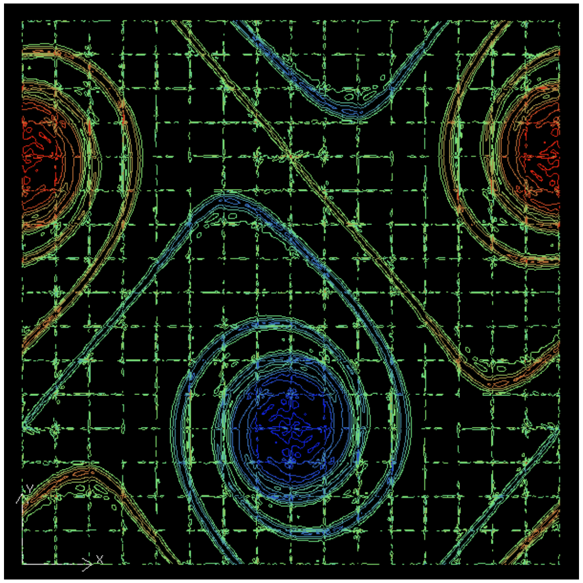
- Filtering acts like well-tuned hyperviscosity
  - Attacks only the fine scale modes (that, numerically speaking, shouldn't have energy anyway…)
  - Can precisely identify which modes in the SE expansion to suppress (unlike differential filters)
  - Does not compromise spectral convergence
- Dealiasing of convection operator recommended for high Reynolds number applications to avoid spurious eigenvalues
  - Can run double shear-layer roll-up problem *forever* with
    - $\nu = 0$,
    - *no filtering*

Dealiased Shear Layer Roll-Up Problem, 128^2

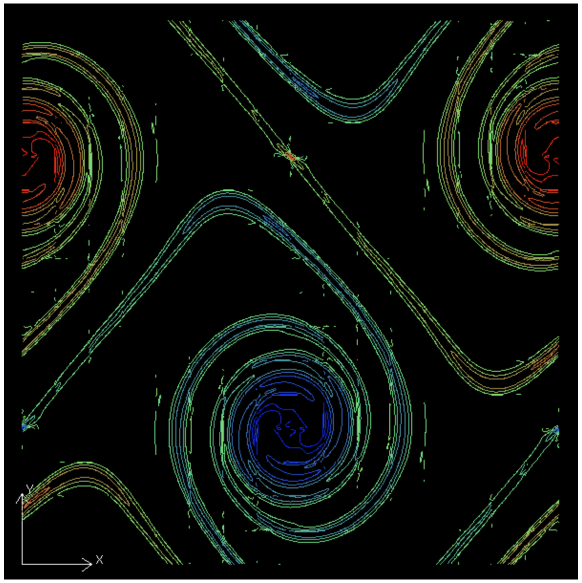
n = 0, no filter



n = 10^{-5}, no filter



n = 0, filter = (.1,.025)

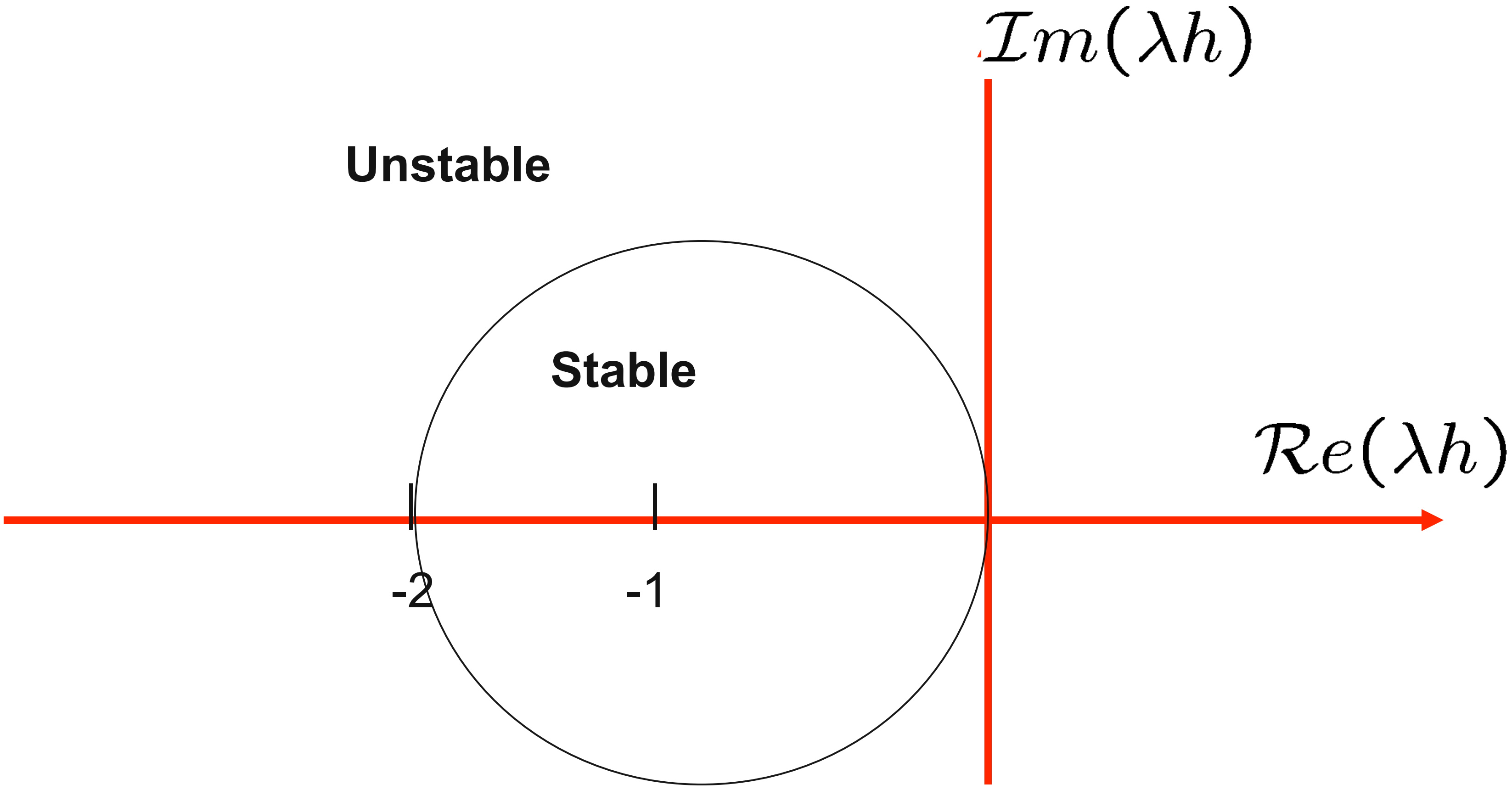


However, Johan Malm established that we do eventually get blow-up with the case on the left!

Thank you!

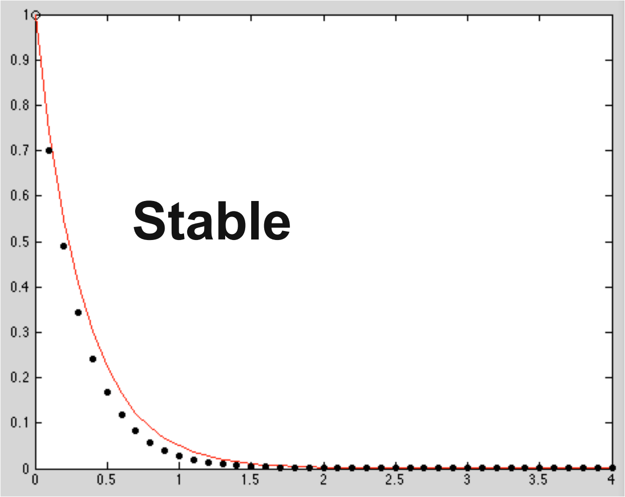
Time for Questions!

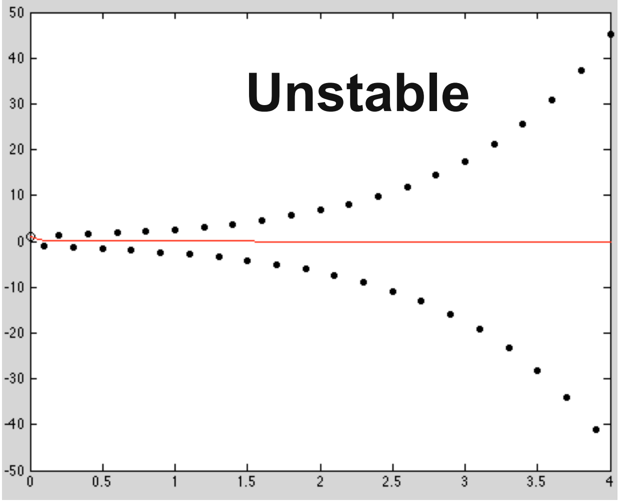
Stability Region for Euler’s Method



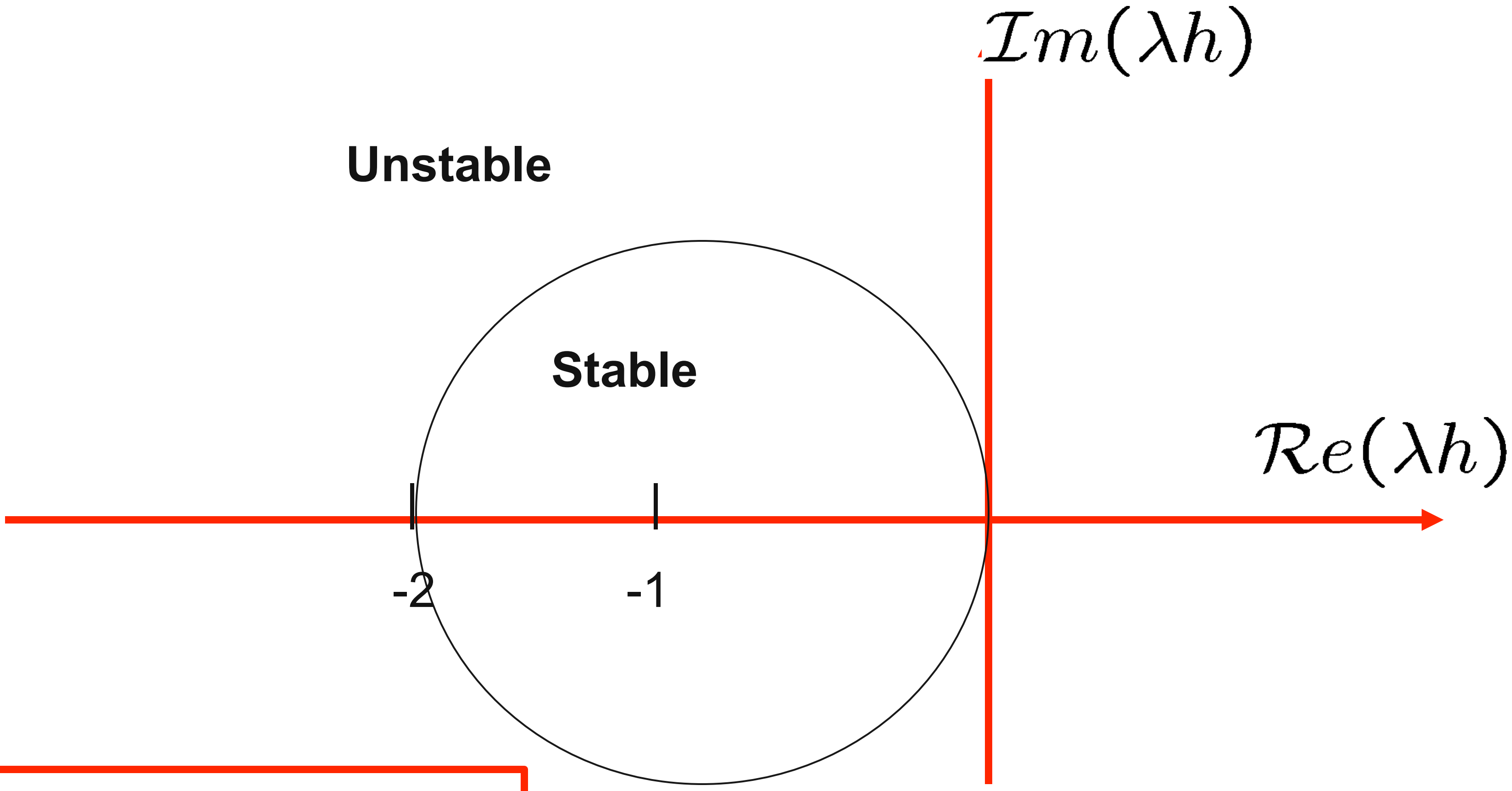
MATLAB EXAMPLE: Euler for y' = λ y (ef1.m)

```
%% A simple Euler forward integrator
%
%  Typical Usage:  h=.01; lambda=3; ef1
%

tfinal = 4; nsteps=ceil(tfinal/h); h=tfinal/nsteps;

x=zeros(nsteps+1,1);t=x;

t=h*(0:nsteps);

hold off; x(1)=1; plot(t(1),x(1),'ko'); hold on;

xe=x(1)*exp(lambda*t); plot(t,xe,'r-')

for k=1:nsteps;

    fx = lambda*x(k);

    x(k+1)=x(k) + h*fx;
    t(k+1)=k*h;

    plot(t(k+1),x(k+1),'k.'); drawnow;

end;
```





Stability Region for Euler's Method



Why complex plane?

Recall: Orbit Example

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = A\,\mathbf{y}.$$

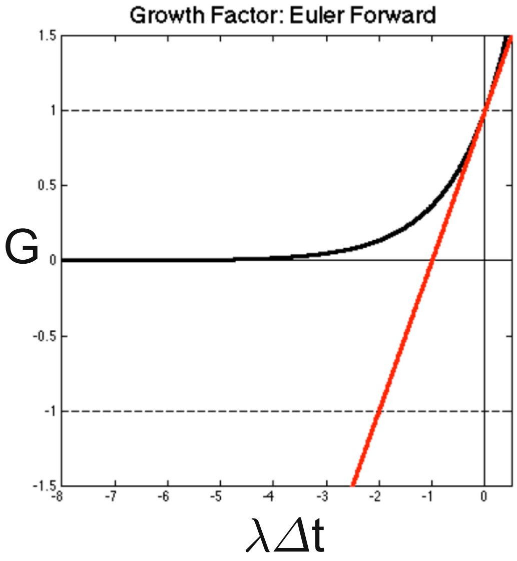
$$\frac{d\mathbf{y}}{dt} = A\mathbf{y}$$

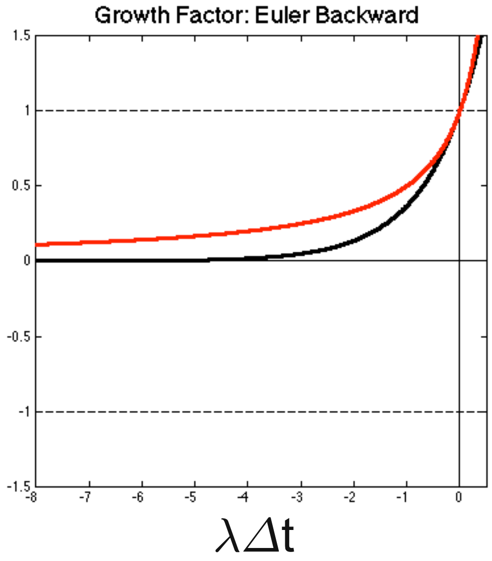
$$\begin{aligned} \left| A - \lambda I \right| &= \begin{vmatrix} -\lambda & -1 \\ 1 & -\lambda \end{vmatrix} \\ &= \lambda^2 + 1 = 0 \end{aligned}$$

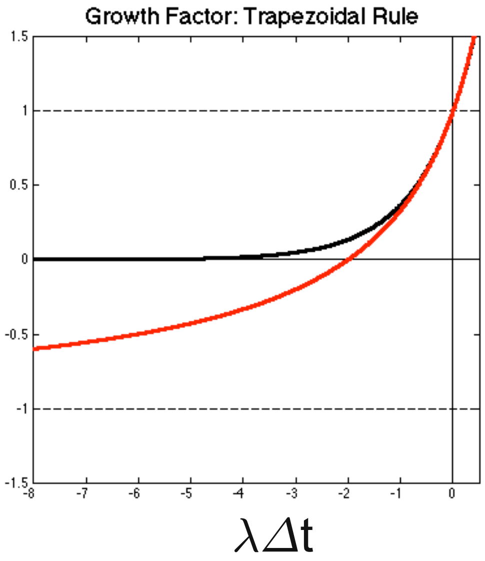
$$\lambda = \pm i$$

- Even though ODE involves only reals, the behavior can be governed by complex eigenvalues.

Growth Factors for Real λ







- Each growth factor approximates $e^{\lambda\Delta t}$ for $\lambda\Delta t$ → 0
- For EF, |G| is not bounded by 1
- For Trapezoidal Rule, local (small$\lambda\Delta t$) approximation is O($\lambda\Delta t^2$), but |
G| → -1 as $\lambda\Delta t$ → $-\infty$.
- BDF2 will give 2nd-order accuracy, stability, and |G|→0 as $\lambda\Delta t$ → $-\infty$.

BDFk Formulas: GTE = O(Δt^k)

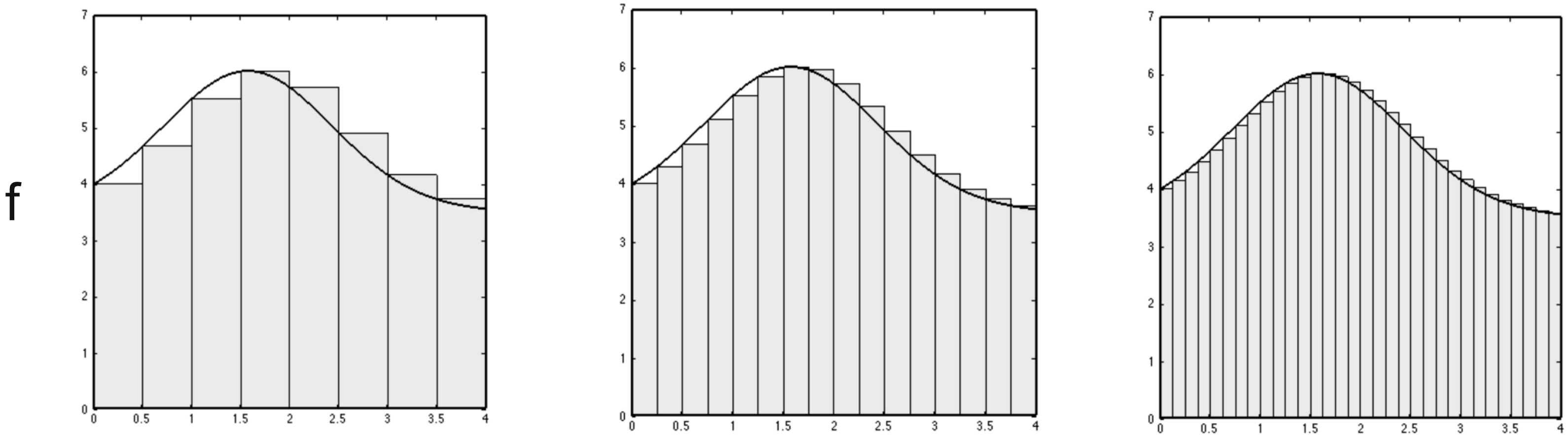
$$\text{BDF1:} \quad \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{u^n - u^{n-1}}{\Delta t} + O(\Delta t)$$

$$\text{BDF2:} \quad \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{3u^n - 4u^{n-1} + u^{n-2}}{2\Delta t} + O(\Delta t^2)$$

$$\text{BDF3:} \quad \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{11u^n - 18u^{n-1} + 9u^{n-2} - 2u^{n-3}}{6\Delta t} + O(\Delta t^3).$$

- Unlike the trapezoidal rule, these methods are L-stable:
  - |G|→0 as $\lambda \Delta$t → -∞
- k-th order accurate
- Implicit
- Unconditionally stable only for k $\leq$ 2
- Multi-step: require data from previous timesteps

Relationship between LTE and GTE



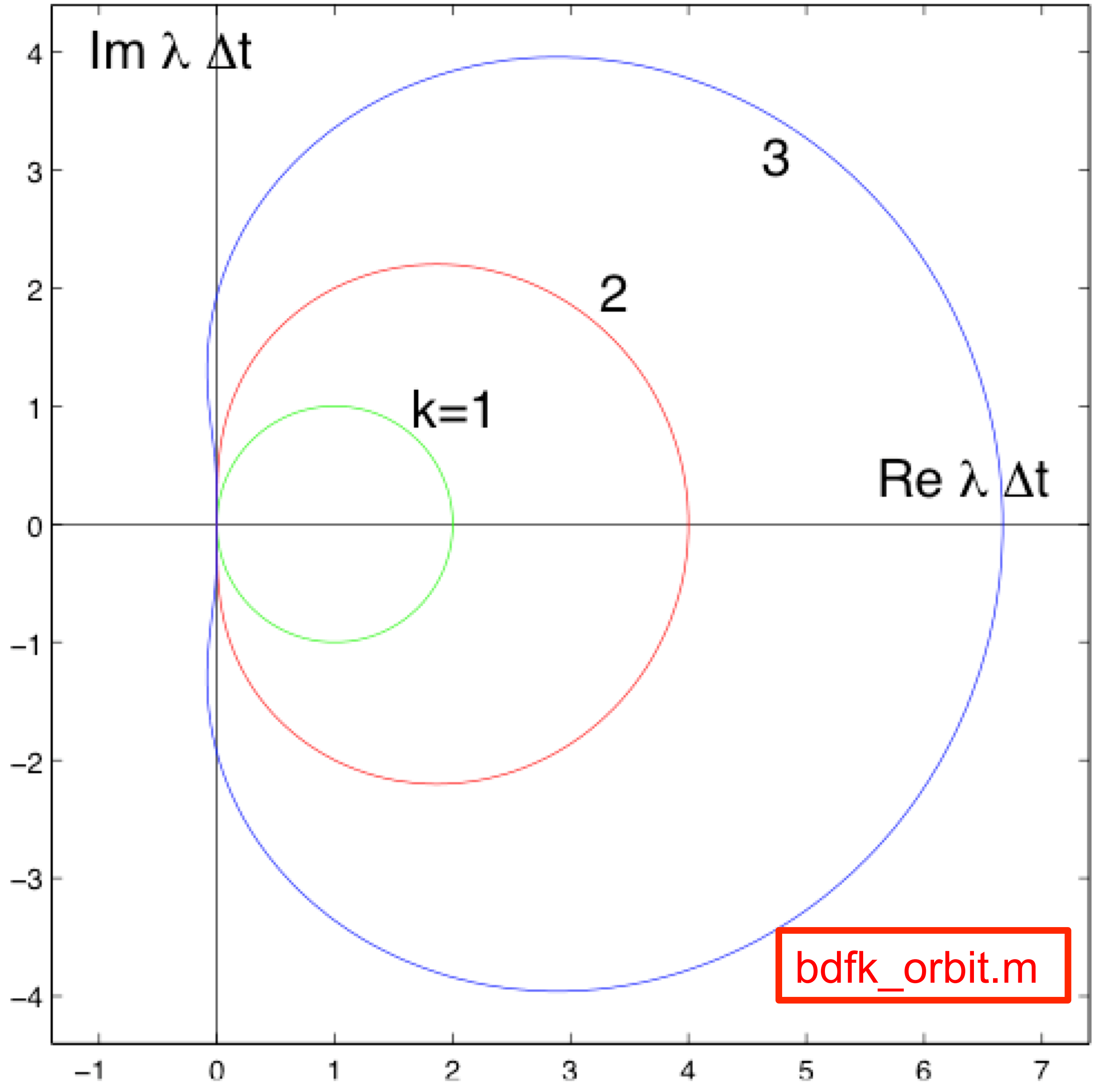
$$y_n \quad = \quad y_0 \; + \int_0^T f(t,y)\,dt$$

- If LTE $= O(\Delta t^2)$, then commit $O(\Delta t^2)$ error on each step.
- Interested in final error at time $t = T = n\Delta t$.
- Interested in the final error $e_n := y(t_n) \; - \; y_n$ in the limit $n \longrightarrow \infty$, $n\Delta t = T$ *fixed.*
- Nominally, the final error will be proportional to the sum of the local errors,

$$e_n \quad \sim \quad C\,n \cdot \text{LTE} \sim C\,n\Delta t^2 \sim C\,(n\Delta t)\Delta t \sim C\,T\Delta t$$

- GTE $\sim$ LTE $/\Delta t$

BDFk Neutral Stability Curve



bdfk_orbit.m

Explicit High-Order Methods

- High-order explicit methods are of interest for several reasons:

  - Lower cost per step than implicit (but possibly many steps if system has disparate timescales, i.e., is stiff --- spring-mass example).

  - More accuracy

  - For k > 2, encompass part of the imaginary axis near zero, so stable for systems having purely imaginary eigenvalues.

  - We'll look at three classes of high-order explicit methods:
    - *BDFk / Ext k*
    - *kth-order Adams Bashforth*
    - *Runge-Kutta methods*
  - Each has pros and cons…

Higher-Order Explicit Timesteppers: BDFk/EXTk

- Idea: evaluate left-hand and right-hand sides at t_{k+1} to accuracy $O(\Delta t^k)$.

$$\left.\frac{dy}{dt}\right|_{t_{k+1}} \quad = \quad f(t,y)|_{t_{k+1}}$$

- Can treat term on the right via kth-order extrapolation.

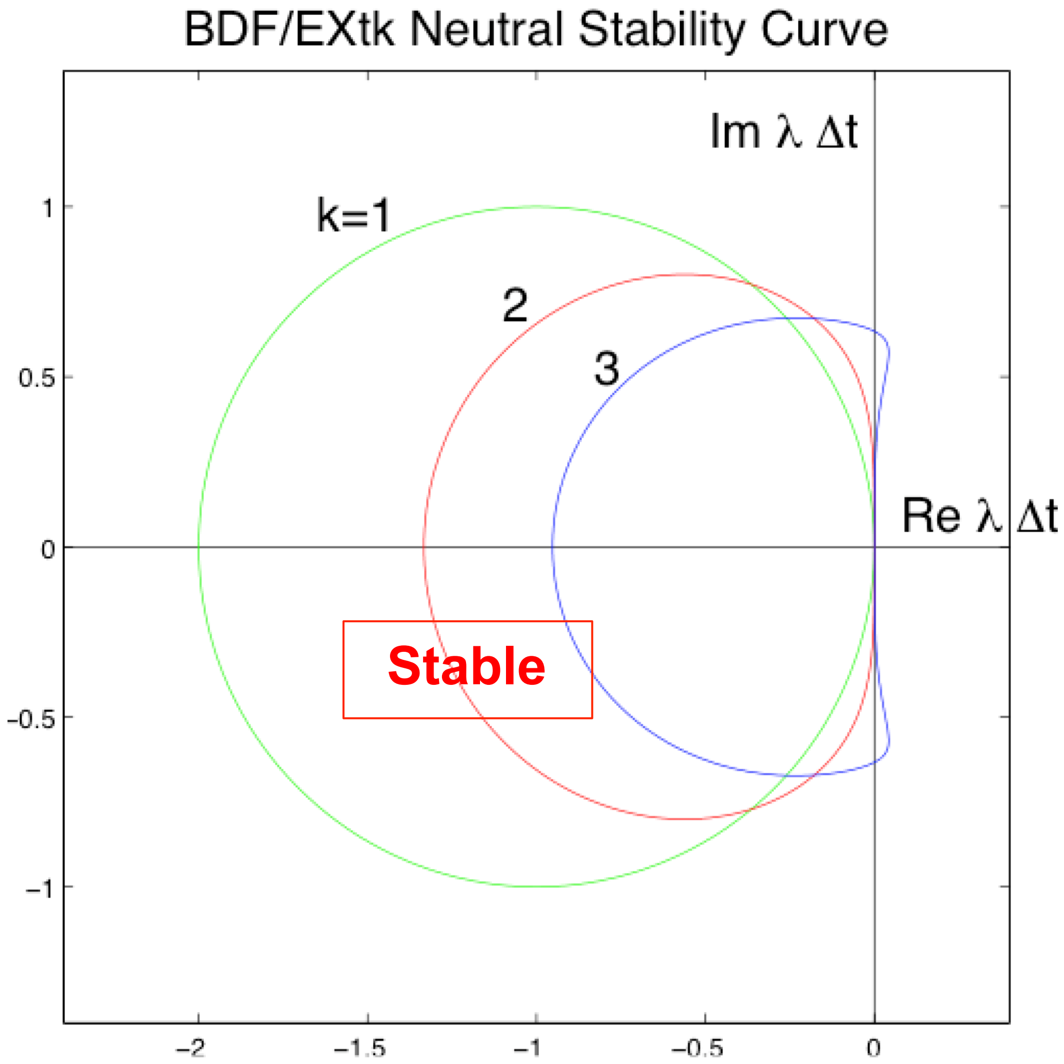
- For example, for $k = 2$,

$$\frac{3y_{k+1} \;-\; 4y_k \;+\; y_{k-1}}{2\Delta t} \;+\; O(\Delta t^2) \;\;=\;\; 2f_k - f_{k-1} \;+\; O(\Delta t^2)$$

- Solve for y_{k+1} in terms of known quantities on the right:

$$y_{k+1} \;\;=\;\; \frac{2}{3}\left[\frac{4y_k \;-\; y_{k-1}}{2} \;+\; \Delta t(2f_k - f_{k-1})\right] \;+\; O(\Delta t^3)$$

- Note that LTE is $O(\Delta t^3)$, GTE=$O(\Delta t^2)$.



- Here we see that the k=3 curve encompasses part of the imaginary axis near the origin of the $\lambda\Delta t$ plane, which is important for stability of non-dissipative systems.

Higher-Order Explicit Timesteppers: kth-order Adams-Bashforth

- Adams-Bashforth methods are a somewhat simpler alternative to BDFk/EXTk.
- Time advancement via integration:

$$\mathbf{y}_{k+1} \;=\; \mathbf{y}_k \;+\; \int_{t_k}^{t_{k+1}} \mathbf{f}(t,\mathbf{y})\,dt$$

- AB1:

$$\int_{t_k}^{t_{k+1}} f(t,\mathbf{y})\,dt \;=\; h_k f_k \;+\; O(h^2)$$

- AB2:

$$\begin{aligned}\int_{t_k}^{t_{k+1}} \mathbf{f}(t,\mathbf{y})\,dt \;&=\; h_k\mathbf{f}_k \;+\; \frac{h_k^2}{2}\left[\frac{\mathbf{f}_k-\mathbf{f}_{k-1}}{h_{k-1}}\right] \;+\; O(h^3)\\ &=\; h\left(\frac{3}{2}\mathbf{f}_k \;-\; \frac{1}{2}\mathbf{f}_{k-1}\right) \;+\; O(h^3)\ \text{(if } h \text{ is constant)}\end{aligned}$$

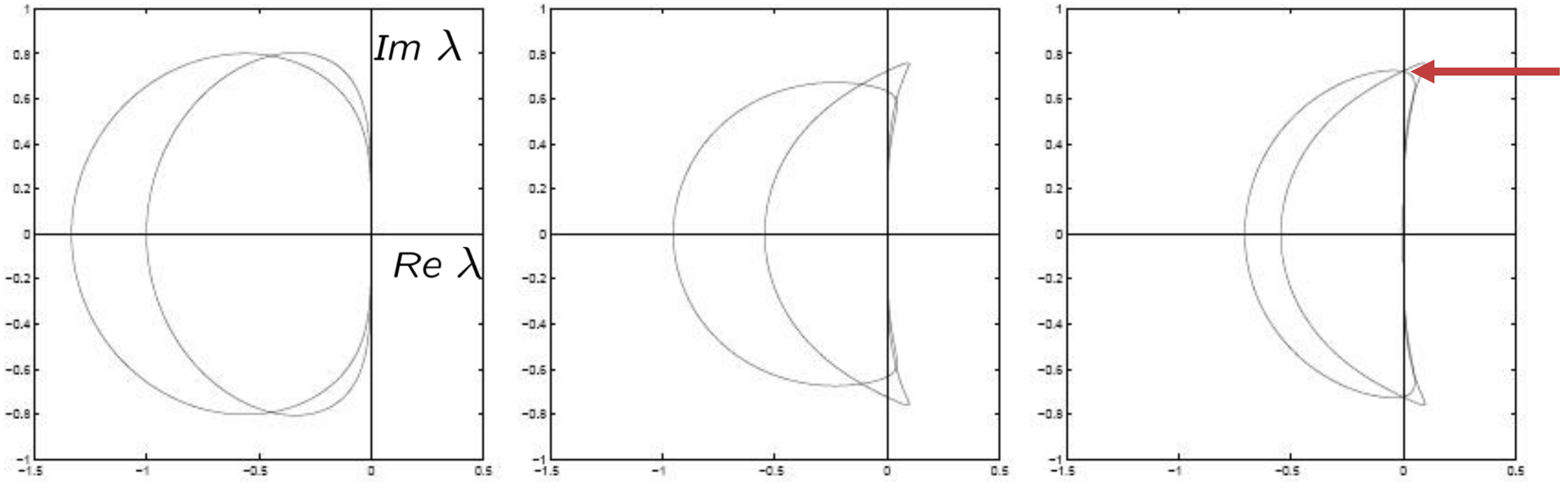
- AB3:

$$\int_{t_k}^{t_{k+1}} \mathbf{f}(t,\mathbf{y})\,dt \;=\; h\left(\frac{23}{12}\mathbf{f}_k \;-\; \frac{16}{12}\mathbf{f}_{k-1} \;+\; \frac{5}{12}\mathbf{f}_{k-2}\right) \;+\; O(h^4)\ \text{(if } h \text{ is constant)}$$

- LTE for ABm is $O(h^{m+1})$. GTE for ABm is $O(h^m)$.

Stability of Various Timesteppers

- Derived from model problem $\frac{du}{dt} = \lambda u$
- Stability regions shown in the $\lambda\Delta t$ plane (stable *inside* the curves)



~ 0.72

Figure 1: Stability regions for (left) AB2 and BDF2/EXT2, (center) AB3 and BDF3/EXT3, and (right) AB3 and BDF2/EXT2a.

- To make effective use of this plot, we need to know something about the eigenvalues λ of the Jacobian.
- But first, *How are these plots generated?*

Determining the Neutral-Stability Curve

Consider BDF2/EXT2, and apply it to $\dfrac{du}{dt} = \lambda u$:

$$3u^m - 4u^{m-1} + u^{m-2} = 2\lambda\Delta t\left(2u^{m-1} - u^{m-2}\right).$$

Seek solutions of the form $u^m = (z)^m$, $z \in C$:

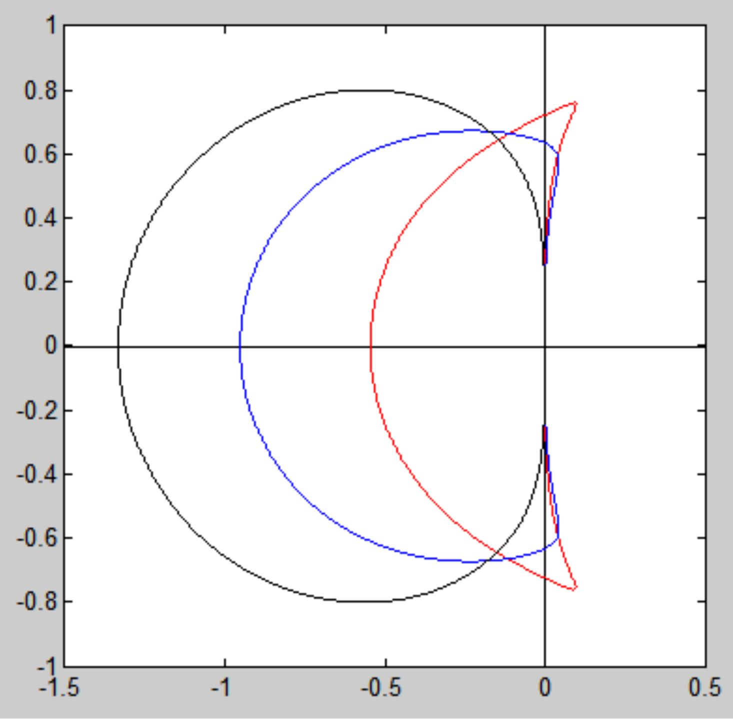
$$\begin{aligned} 3z^m - 4z^{m-1} + z^{m-2} &= 2\lambda\Delta t\left(2z^{m-1} - z^{m-2}\right). \\ 3z^2 - 4z + 1 &= 2\lambda\Delta t\left(2z - 1\right). \end{aligned}$$

Set $z = e^{i\theta}$, $\theta \in [0, 2\pi]$, and solve for $\lambda\Delta t$:

$$\lambda\Delta t = \frac{3e^{i2\theta} - 4e^{i\theta} + 1}{2\left(2e^{i\theta} - 1\right)}.$$

Matlab Code: stab.m

```
ymax=1; ep=1.e-13; yaxis=[-ymax*ii ymax*ii]';  % Plot axes
xaxis=[-2.0+ep*ii 2.0+ep*ii]';
hold off; plot (yaxis,'k-'); hold on; plot (xaxis,'k-');
axis square; axis([-ymax-.5 ymax-.5 -ymax ymax]);

ii=sqrt(-1); th=0:.001:2*pi;  th=th'; ith=ii*th; ei=exp(ith);
E = [ ei 1+0*ei 1./ei 1./(ei.*ei) 1./(ei.*ei.*ei)];


ab0    = [1 0.0 0.0 0. 0.]';
ab1    = [0 1.0 0.0 0. 0.]';
ab2    = [0 1.5 -.5 0. 0.]';
ab3    = [0 23./12. -16./12. 5./12. 0.]';
bdf1   = (([ 1.   -1.   0.   0. 0.])/1.)';
bdf2   = (([ 3.   -4.   1.   0. 0.])/2.)';
bdf3   = (([11. -18.   9. -2. 0.])/6.)';
exm    = [1 0   0 0 0]';
ex1    = [0 1   0 0 0]';
ex2    = [0 2 -1 0 0]';
ex3    = [0 3 -3 1 0]';
du     = [1. -1. 0. 0. 0.]';
```



```
ldtab3 =(E*du)./(E*ab3);   plot (ldtab3 ,'r-');  % AB3
bdf3ex3=(E*bdf3)./(E*ex3); plot (bdf3ex3,'b-');  % BDF3/EXT3
bdf2ex2=(E*bdf2)./(E*ex2); plot (bdf2ex2,'k-');  % BDF2/EXT2
```